

[illegible]

```

LL      NN      NN      KK      KK      000000      PPPPPPPP      TTTTTTTTTT      IIIIII      000000      NN      NN
LL      NN      NN      KK      KK      000000      PPPPPPPP      TTTTTTTTTT      IIIIII      000000      NN      NN
LL      NN      NN      KK      KK      00      00      PP      PP      TT      TT      II      00      00      NN      NN
LL      NN      NN      KK      KK      00      00      PP      PP      TT      TT      II      00      00      NN      NN
LL      NNNN      NN      KK      KK      00      00      PP      PP      TT      TT      II      00      00      NNNN      NN
LL      NNNN      NN      KK      KK      00      00      PP      PP      TT      TT      II      00      00      NNNN      NN
LL      NN      NN      KKKKKK      00      00      PPPPPPPP      TT      TT      II      00      00      NN      NN
LL      NN      NN      KKKKKK      00      00      PPPPPPPP      TT      TT      II      00      00      NN      NN
LL      NN      NNNN      KK      KK      00      00      PP      TT      TT      TT      II      00      00      NN      NNNN
LL      NN      NNNN      KK      KK      00      00      PP      TT      TT      TT      II      00      00      NN      NNNN
LL      NN      NN      KK      KK      00      00      PP      TT      TT      TT      II      00      00      NN      NN
LL      NN      NN      KK      KK      00      00      PP      TT      TT      TT      II      00      00      NN      NN
LLLLLLLLLLLL      NN      NN      KK      KK      000000      PP      TT      TT      IIIIII      000000      NN      NN
LLLLLLLLLLLL      NN      NN      KK      KK      000000      PP      TT      TT      IIIIII      000000      NN      NN

```

```

LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```



```
0001 0 module lnk_procoptions
0002 0      (ident = 'V04-000'
0003 0      ,addressing_mode
0004 0      (external = general
0005 0      ,nonexternal = long_relative
0006 0      )
0007 1 begin
0008 1
0009 1 %title 'Linker options parser'
0010 1
0011 1
0012 1
0013 1 *****
0014 1 *
0015 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0016 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0017 1 *  ALL RIGHTS RESERVED.
0018 1 *
0019 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0020 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0021 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0022 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0023 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0024 1 *  TRANSFERRED.
0025 1 *
0026 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0027 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0028 1 *  CORPORATION.
0029 1 *
0030 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0031 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0032 1 *
0033 1 *
0034 1 *****
0035 1
0036 1
0037 1
0038 1
0039 1
0040 1
0041 1 ++
0042 1
0043 1 MODULE: LNK_PROCOPTIONS
0044 1
0045 1 FACILITY: LINKER
0046 1
0047 1 ABSTRACT: Options processing
0048 1
0049 1 HISTORY:
0050 1
0051 1     VERSION: V03-001
0052 1
0053 1     AUTHOR: B. Schreiber, 10-Jul-1980
0054 1
0055 1     MODIFIED BY:
0056 1
0057 1     V03-33 ADE0003      Alan D. Eldridge      13-Aug-1984
```



```
58 0058 1 Always save options file input in an OEB. It was only
59 0059 1 being done if LNK$V_LONG was set, but that flag doesn't
60 0060 1 get set until after the routines in this module have executed.
61 0061 1
62 0062 1 V03-032 ADE0002 Alan D. Eldridge 6-Aug-1984
63 0063 1 Add LNK$GL_UNSUPPORTED support. This is a mask settable
64 0064 1 from the options file, and is used at this time solely to
65 0065 1 enable demand zero ISD's in shareable images.
66 0066 1
67 0067 1 V03-031 ADE0001 Alan D. Eldridge 29-Jun-1984
68 0068 1 Remove improper complaint about DZRO_MIN option being
69 0069 1 ignored for shareable images.
70 0070 1
71 0071 1 V03-030 JWT0156 Jim Teague 20-Feb-1984
72 0072 1 The SYMBOL option and the UNIVERSAL option don't
73 0073 1 play together very well. Declaring a symbol
74 0074 1 by the SYMBOL option followed by making it
75 0075 1 UNIVERSAL would yield an undefined symbol.
76 0076 1 However, reversing the order would cause an
77 0077 1 access violation.
78 0078 1
79 0079 1 V03-029 JWT0149 Jim Teague 21-Dec-1983
80 0080 1 Allow the characters "$" and "_" in directory specs.
81 0081 1
82 0082 1 V03-028 MCN0001 Maria del C. Nasr 25-Jul-1983
83 0083 1 If SET VERIFY is on and option file is being read from
84 0084 1 SYS$INPUT, do not echo line to output. DCL is doing
85 0085 1 that now.
86 0086 1
87 0087 1 V03-027 JWT0125 Jim Teague 21-Jun-1983
88 0088 1 Reincarnate UNIV=* when accompanied by /NOEXE.
89 0089 1
90 0090 1 V03-026 JWT0123 Jim Teague 27-May-1983
91 0091 1 Add message for UNIV=*.
92 0092 1
93 0093 1 V03-025 JWT0121 Jim Teague 20-May-1983
94 0094 1 Sound the death knell for UNIVERSAL=*.
95 0095 1
96 0096 1 V03-024 JWT0108 Jim Teague 13-Mar-1983
97 0097 1 Adjustments to new CLI interface.
98 0098 1
99 0099 1 V03-023 JWT0099 Jim Teague 14-Mar-1983
100 0100 1 New CLI interface.
101 0101 1
102 0102 1 V03-022 JWT0071 Jim Teague 02-Dec-1982
103 0103 1 Added NAME and IDENTIFICATION options.
104 0104 1
105 0105 1 V03-021 JWT0050 Jim Teague 10-Aug-1982
106 0106 1 Add IMAGE_TYPE option with CLI value.
107 0107 1
108 0108 1 V03-020 JWT0034 Jim Teague 25-May-1982
109 0109 1 Remove the GSMATCH "NEVER" option.
110 0110 1
111 0111 1 --
112 0112 1
113 0113 1 %sbttl 'Declarations'
114 0114 1
```


115	0115	1	!	
116	0116	1	!	Include files:
117	0117	1	!	
118	0118	1	!	
119	0119	1	!	Library 'LIBL32';
120	0120	1	!	!System macros
121	0121	1	!	Library 'TPAMACL32';
122	0122	1	!	!TPARSE macros
123	0123	1	!	Library 'DATBAS';
124	0124	1	!	!Linker data structures
125	0125	1	!	require 'PREFIX';
126	0240	1	!	!Linker macros
127	0241	1	!	forward routine
128	0242	1	!	readoption,
129	0243	1	!	!Read record from options file
130	0244	1	!	set_alluniv,
131	0245	1	!	!Set all universal temporarily
132	0246	1	!	scanline,
133	0247	1	!	!Collapse continuation lines
134	0248	1	!	set_base,
135	0249	1	!	!Set image base
136	0250	1	!	set_channels,
137	0251	1	!	!Set number of I/O channels
138	0252	1	!	set_unsupported,
139	0253	1	!	!Set 'unsupported features' mask
140	0254	1	!	insertcluster,
141	0255	1	!	!Insert cluster in cluster list
142	0256	1	!	processfile,
143	0257	1	!	!Process a file specification
144	0258	1	!	getfilename,
145	0259	1	!	!Extract filename portion of file specification
146	0260	1	!	search_insert_symbol,
147	0261	1	!	!Lookup/insert a symbol into symbol table
148	0262	1	!	crossref_symbol,
149	0263	1	!	!Enter symbol into cross reference
150	0264	1	!	set_universal,
151	0265	1	!	!Process universal symbols
152	0266	1	!	definesymbolname,
153	0267	1	!	!Process symbol name part of defining a symbol
154	0268	1	!	definesymbolval,
155	0269	1	!	!Process value part of defining a symbol
156	0270	1	!	createpsect,
157	0271	1	!	!Create a psect
158	0272	1	!	createcluster,
159	0273	1	!	!Create a cluster
160	0274	1	!	setclusterbase,
161	0275	1	!	!Set the base address of a cluster
162	0276	1	!	setclusterpfc,
163	0277	1	!	!Set page fault factor for cluster
164	0278	1	!	clusterdone,
165	0279	1	!	!Finish cluster option processing
166	0280	1	!	set_collect,
167	0281	1	!	!Start collecting psects into clusters
168	0282	1	!	collect_psect,
169	0283	1	!	!Collect a psect into a cluster
170	0284	1	!	set_gsmatch_ctl,
171	0285	1	!	!Store match control part of GSMATCH option
			!	set_gsmatch_maj,
			!	!Store major id of GSMATCH
			!	set_gsmatch_min,
			!	!Store minor id of GSMATCH
			!	set_min_dzro,
			!	!Store dzro_min option data
			!	set_isd_max,
			!	!Store isd_max option data
			!	set_ioseg,
			!	!Set size of I/O segment
			!	set_pscatib,
			!	!Set psect attributes
			!	set_p0bufs,
			!	!Set/clear nop0bufs flag
			!	set_protect,
			!	!Set protect flag
			!	set_shl,
			!	!Set shl_extra option
			!	set_stack,
			!	!Set stack option
			!	set_include,
			!	!Set /include qualifier
			!	set_library,
			!	!Set /library qualifier
			!	set_selective,
			!	!Set /selective qualifier
			!	set_shareable,
			!	!Set /shareable qualifier
			!	set_shrcopyflag,
			!	!Set copy / nocopy flag for /shareable qualifier
			!	set_qual_flags,
			!	!Set bits for qualifiers
			!	set_image_type,
			!	!Set image type bit
			!	set_image_ident,
			!	!Set image file identification
			!	set_image_name,
			!	!Set image name
			!	search_insert_list,
			!	!Search linked list for name
			!	namelengthcheck,
			!	!Check max length of name
			!	debug_stop,
			!	!Stop to look at tparse tables

```
172 0286 1 missingargerr,
173 0287 1 optionvaluerr,
174 0288 1 syntaxerr;
175 0289 1
176 0290 1
177 0291 1 External references
178 0292 1
179 0293 1 external routine
180 0294 1 inputfile,
181 0295 1 crf$insrtref,
182 0296 1 crf$insrtkey,
183 0297 1 lib$insert_tree,
184 0298 1 lib$lookup_tree,
185 0299 1 lnk$alloblk,
186 0300 1 lnk$alloc_pdd,
187 0301 1 lnk$alloccluster,
188 0302 1 lnk$compare_pdd,
189 0303 1 lnk$clunamcmp,
190 0304 1 lnk$insert_clu,
191 0305 1 lnk$dealblk,
192 0306 1 lnk$insudfsym,
193 0307 1 lnk$insert,
194 0308 1 lnk$search,
195 0309 1 lnk$setlibrin,
196 0310 1 lnk$setshrblin,
197 0311 1 lnk$upcase_d,
198 0312 1 lib$put_output,
199 0313 1 lib$parse,
200 0314 1 sys$fao;
201 0315 1
202 0316 1 external
203 0317 1 lnk$gb_matchctl : byte,
204 0318 1 lnk$gl_matchid : bblock,
205 0319 1 lnk$gl_defclu : bblock,
206 0320 1 lnk$gl_curclu : ref bblock,
207 0321 1 lnk$gl_clutree,
208 0322 1 lnk$gl_ctlmsk : bblock,
209 0323 1 lnk$gl_fvmlst : ref bblock,
210 0324 1 lnk$gw_stack : word,
211 0325 1 lnk$gt_pscstring,
212 0326 1 lnk$gt_clustrng,
213 0327 1 lnk$gt_symstring,
214 0328 1 lnk$gt_imgnam : vector [, byte],
215 0329 1 lnk$gt_imgid : vector [, byte],
216 0330 1 lnk$al_valctlb,
217 0331 1 lnk$al_sytblfmt,
218 0332 1 lnk$gl_maxsymsz,
219 0333 1 lnk$gl_maxmodsz,
220 0334 1 lnk$gl_sysinput_flag,
221 0335 1 lnk$gw_ncrosrfs : word,
222 0336 1 lnk$gw_misects : word,
223 0337 1 lnk$gw_dzromin : word,
224 0338 1 lnk$gw_nudfsyms : word;
225 0339 1
226 0340 1 external literal
227 0341 1 lin$cmdtoolong,
228 0342 1 lin$confqual,
```

```
!Argument missing for option
!Value out of range error
!Report syntax error

!Process input file
!Insert cross reference reference
!Insert key into cross reference
!Insert into balanced binary tree
!Lookup in balanced binary tree
!Allocate dynamic memory
!Allocate psect def. descriptor
!Allocate a cluster descriptor
!Compare names in psect def. descriptor
!Compare cluster name routine
!Insert cluster into cluster tree
!Deallocate dynamic memory
!Insert a symbol into the undefined list
!Insert a symbol into symbol table
!Search symbol table for symbol
!Set input file as a library
!Set input file as a shareable image
!Convert string to upper case
!Write to SYSS$OUTPUT
!Table driver parser
!Formatted ASCII output

!Global section match control
!Global section match
!Default cluster descriptor
!Pointer to current cluster descriptor
!Tree head for cluster tree
!Linker control flags
!Listhead of free VM
!Size of stack
!ASCII string that says PSECT
!or cluster
!or string
!Image name field
!Image file identification field
!Cref by value control table
!Cref by symbol control table
!Maximum symbol name length
!Maximum module name length
!Set if option file is SYSS$INPUT
!Number of cross references
!Maximum number of isects
!Minimum demand zero number of pages
!Number of undefined symbols

!Command line too long
!Conflicting qualifiers
```



```

229 0343 1 lin$_crferr, !Error from cross reference
230 0344 1 lin$_illnamelen, !Illegal name length
231 0345 1 lin$_linerr, !Command segment in error
232 0346 1 lin$_mulcluopt, !Cluster multiply defined in options file
233 0347 1 lin$_optignshr, !Option ignored in shareable image
234 0348 1 lin$_optignsys, !Option ignored in system image
235 0349 1 lin$_optlin, !Option line in error
236 0350 1 lin$_optsynerr, !General syntax error
237 0351 1 lin$_optvalerr, !Option value error
238 0352 1 lin$_optargmis, !Argument missing for option
239 0353 1 lin$_premeof, !Premature eof on options file
240 0354 1 lin$_readerr, !Read error on file blah
241 0355 1 lin$_shrcpyign, !/SHARE=COPY attempted
242 0356 1 lin$_shrinsys, !Shareable image in system image
243 0357 1 lin$_shrsepclu, !Shareable image requires seperate cluster
244 0358 1 lnk$_k_max_filename_length; !Maximum filename length
245 0359 1
246 0360 1 ! Own storage
247 0361 1
248 0362 1 global
249 0363 1 lnk$_gl_inclst : ref bblock [dsc$_c_s_bln], !Include list descriptor
250 0364 1 lnk$_gl_defclunum, !Default cluster number
251 0365 1 lnk$_gl_optextp : ref bblock, !Listhead of option file text
252 0366 1 lnk$_gl_optexte : ref bblock initial (lnk$_gl_optextp), !Pointer to last text block
253 0367 1 lnk$_gl_shlextra, !Requested shl_extra
254 0368 1 lnk$_gw_chans : word, !Number of channels requested
255 0369 1 lnk$_gw_ioseg : word, !Size of i/o segment
256 0370 1 lnk$_gq_privs : vector [2] initial (long (-1, -1)), !Requested privileges
257 0371 1 lnk$_gl_unsupported : long initial (0), !Unsupported features mask
258 0372 1 lnk$_gl_cclulst : ref bblock, !List head for collect cluster list
259 0373 1 lnk$_gl_pscdflst : ref bblock; !List head for psect definitions
260 0374 1
261 0375 1 global literal
262 0376 1 lnk$_k_maxchans = 4095; !Maximum number of channels
263 0377 1 lnk$_k_maxstack = 65535; !Maximum stack pages
264 0378 1 lnk$_k_maxioseg = 65535; !Maximum size of I/O segment
265 0379 1 lnk$_k_maxisds = 65535; !Maximum number of isects
266 0380 1 lnk$_k_mindzro = 65535; !Maximum min_dzro size
267 0381 1 lnk$_k_maxmajid = 255; !Maximum major ident
268 0382 1 lnk$_k_maxminid = %x'FFFFFF', !Maximum minor ident
269 0383 1 lnk$_k_maxoptlin = 32767; !Maximum size of option line
270 0384 1 lnk$_k_maxpfc = 255; !Maximum page fault cluster factor
271 0385 1
272 0386 1 own
273 0387 1 tparse_block : bblock [tpa$_k_length0] !Tparse parameter block
274 0388 1 initial (long (tpa$_k_count0)
275 0389 1 , long (tpa$_m_abbrfm) !Abbreviate first match
276 0390 1 );
277 0391 1 cmdbuffer, !Address of command buffer
278 0392 1 optlinedesc : bblock [dsc$_c_s_bln], !String descriptor of complete line
279 0393 1 fileflagsadr : ref bblock, !Address of caller's file flags
280 0394 1 optrabadr : ref bblock, !Address of RAB for GET sequential
281 0395 1 include_desc : bblock [dsc$_c_s_bln], !Address of include list
282 0396 1 curcollectdesc : ref bblock, !Address of current collect descriptor
283 0397 1 curpsectdesc : ref bblock, !Address of current psect descriptor
284 0398 1 cursymdesc : ref bblock, !Address of current symbol descriptor
285 0399 1 cursymsnb : ref bblock, !Address of current symbol name block

```

```

: 286      0400 1 filespec_desc : bblock [dsc$_s_bln],      !String descriptor for file specification
: 287      0401 1 optionfilename : ref bblock,              !Address of string descriptor for options file name
: 288      0402 1 quote_flag,                                !True when inside double quotes
: 289      0403 1 cluoptflag,                                !True if processing cluster option
: 290      0404 1 sharecopy,                                !Set true if /share=copy
: 291      0405 1 protectflag;                               !set true if clusters are to be protected
: 292      0406 1
: 293      0407 1 literal
: 294      0408 1     delimited = 1,                        !Ident string is delimited
: 295      0409 1     sign_bit  = %x'80000000',            !Mask for sign bit
: 296      0410 1     word_sign_bit = %x'8000',            !Sign bit of a word
: 297      0411 1     rh_mask    = %x'FFFF',              !Mask for right half
: 298      0412 1
: 299      0413 1 bind
: 300      0414 1     image_ident_name = cstring ('Image identification'),
: 301      0415 1     image_name_name  = cstring ('Image name'),
: 302      0416 1     channels_name    = cstring ('Channel'), !Name of option
: 303      0417 1     clusterpf_name  = cstring ('Page fault cluster'),
: 304      0418 1     defined_by_option = cstring ('<Linker option>'),
: 305      0419 1     optionfilestring = cstring ('OPTION FILE'),
: 306      0420 1     dzromin_name     = cstring ('DZRO_MIN'),
: 307      0421 1     gsmatch_name     = cstring ('GSMATCH'),
: 308      0422 1     ioseg_name       = cstring ('IOSEG'),
: 309      0423 1     isdmax_name      = cstring ('ISD_MAX'),
: 310      0424 1     unsupported_name = cstring ('Unsupported'),
: 311      0425 1     stack_name       = cstring ('Stack');
: 312      0426 1

```



```

: 314      0427 1 %sbttl 'Options parser TPARSE table';
: 315      0428 1
: 316      0429 1 Linker options command parsing table
: 317      0430 1
: 318      0431 1 $init_state (state_table, key_table);
: 319      0432 1
: 320      0433 1 Parse Linker options
: 321      0434 1
: 322      P 0435 1 $state (option, ((parse_filelist), tpa$_exit),
: 323      P 0436 1 ((base_option), tpa$_exit),
: 324      P 0437 1 ((channels_option), tpa$_exit),
: 325      P 0438 1 ((cluster_option), tpa$_exit),
: 326      P 0439 1 ((collect_option), tpa$_exit),
: 327      P 0440 1 ((dzro_min_option), tpa$_exit),
: 328      P 0441 1 ((gsmatch_option), tpa$_exit),
: 329      P 0442 1 ((iosegment_option), tpa$_exit),
: 330      P 0443 1 ((isd_max_option), tpa$_exit),
: 331      P 0444 1 ((protect_option), tpa$_exit),
: 332      P 0445 1 ((psect_option), tpa$_exit),
: 333      P 0446 1 ((shl_option), tpa$_exit),
: 334      P 0447 1 ((stack_option), tpa$_exit),
: 335      P 0448 1 ((symbol_option), tpa$_exit),
: 336      P 0449 1 ((universal_option), tpa$_exit),
: 337      P 0450 1 ((unsupported_option), tpa$_exit),
: 338      P 0451 1 ((image_type_option), tpa$_exit),
: 339      P 0452 1 ((image_ident_option), tpa$_exit),
: 340      0453 1 ((image_name_option), tpa$_exit));
: 341      0454 1
: 342      0455 1 Parse BASE option
: 343      0456 1
: 344      P 0457 1 $state (base_option,
: 345      0458 1 ('BASE'));
: 346      P 0459 1 $state (
: 347      0460 1 ('=', baseop1));
: 348      P 0461 1 $state (baseop1,
: 349      0462 1 ((parse_number), baseop2, set_base), ((errorparse_1)));
: 350      P 0463 1 $state (baseop2,
: 351      0464 1 ((endline_1), tpa$_exit));
: 352      0465 1
: 353      0466 1 Parse CHANNELS option
: 354      0467 1
: 355      P 0468 1 $state (channels_option,
: 356      0469 1 ('CHANNELS'));
: 357      P 0470 1 $state (
: 358      0471 1 ('=', chanop1));
: 359      P 0472 1 $state (chanop1,
: 360      0473 1 ((parse_number), chanop2, set_channels), ((errorparse_1)));
: 361      P 0474 1 $state (chanop2,
: 362      0475 1 ((endline_1), tpa$_exit));
: 363      0476 1
: 364      0477 1 Parse CLUSTER option
: 365      0478 1
: 366      P 0479 1 $state (cluster_option,
: 367      0480 1 ('CLUSTER'));
: 368      P 0481 1 $state (
: 369      0482 1 ('=');
: 370      P 0483 1 $state (

```

```
371 P 0484 1 ((get_symbol), cluop1, createcluster), !Cluster name
372 0485 1 ((errorparse_1));
373 P 0486 1 $state (cluop1,
374 0487 1 (tpa$eos, cluop7a), !If that's all, then that's ok
375 0488 1 (' ', cluop2), ((errorparse_1));
376 P 0489 1 $state (cluop2,
377 0490 1 (tpa$eos, cluop7a), !If that's all, then that's ok
378 0491 1 ((parse_number_or_null), cluop3, setclusterbase), !Cluster base address
379 0492 1 ((errorparse_1));
380 P 0493 1 $state (cluop3,
381 0494 1 (tpa$eos, cluop7a), !If that's all, then that's ok
382 0495 1 (' ', cluop4), ((errorparse_1));
383 P 0496 1 $state (cluop4,
384 0497 1 (tpa$eos, cluop7a), !If that's all, then that's ok
385 0498 1 ((parse_number_or_null), cluop5, setclusterpfc), !Cluster page fault cluster factor
386 0499 1 ((errorparse_1));
387 P 0500 1 $state (cluop5,
388 0501 1 (tpa$eos, cluop7a), !Allow null file list
389 0502 1 (' ', cluop6), ((errorparse_1));
390 P 0503 1 $state (cluop6,
391 0504 1 (tpa$eos, cluop7a), !Allow null file list
392 0505 1 ((parse_filelist), cluop7), !List of files in this cluster
393 0506 1 ((errorparse_1));
394 P 0507 1 $state (cluop7,
395 0508 1 ((endline_1), cluop8));
396 P 0509 1 $state (cluop7a, !No files in cluster option
397 0510 1 (tpa$lambda, cluop8, insertcluster));
398 P 0511 1 $state (cluop8,
399 0512 1 (tpa$lambda, tpa$exit, clusterdone));
400 0513 1
401 0514 1 Parse COLLECT option
402 0515 1
403 P 0516 1 $state (collect_option,
404 0517 1 ('COLLECT'));
405 P 0518 1 $state (
406 0519 1 ('='));
407 P 0520 1 $state (
408 0521 1 ((get_symbol), collop1, set_collect), !Cluster name
409 0522 1 ((errorparse_1));
410 P 0523 1 $state (collop1,
411 0524 1 (' ', collop2), ((errorparse_1));
412 P 0525 1 $state (collop2,
413 0526 1 ((collect_psect_list), collop3), ! followed by a list of psects
414 0527 1 ((errorparse_1));
415 P 0528 1 $state (collop3,
416 0529 1 ((endline_1), tpa$exit));
417 0530 1
418 0531 1 Parse the psect list in the COLLECT option
419 0532 1
420 P 0533 1 $state (collect_psect_list,
421 0534 1 ((get_symbol), colpsc1, collect_psect), !Get a psect name
422 0535 1 ((errorparse_1));
423 P 0536 1 $state (colpsc1,
424 0537 1 (' ', collect_psect_list), (tpa$eos, tpa$exit), ((errorparse_1));
425 0538 1
426 0539 1 Parse DZRO_MIN option
427 0540 1
```



```

: 428 P 0541 1 $state (dzro_min_option,
: 429 0542 1 ('DZRO_MIN'));
: 430 P 0543 1 $state (
: 431 0544 1 ('=', dzromin1));
: 432 P 0545 1 $state (dzromin1,
: 433 0546 1 ((parse_number), dzromin2, set_min_dzro), ((errorparse_1)));
: 434 P 0547 1 $state (dzromin2,
: 435 0548 1 ((endline_1), tpa$_exit));
: 436 0549 1
: 437 0550 1 Parse GSMATCH option
: 438 0551 1
: 439 P 0552 1 $state (gsmatch_option,
: 440 0553 1 ('GSMATCH'));
: 441 P 0554 1 $state (
: 442 0555 1 ('='));
: 443 P 0556 1 $state (
: 444 0557 1 ((gsmatch_parse_control), gsmop1), !GSMATCH control
: 445 0558 1 ((errorparse_1)));
: 446 P 0559 1 $state (gsmop1,
: 447 0560 1 ('', gsmop2), ((errorparse_1)));
: 448 P 0561 1 $state (gsmop2,
: 449 0562 1 ((parse_number), gsmop3, set_gsmatch_maj), !Major id
: 450 0563 1 ((errorparse_1)));
: 451 P 0564 1 $state (gsmop3,
: 452 0565 1 ('', gsmop4), ((errorparse_1)));
: 453 P 0566 1 $state (gsmop4,
: 454 0567 1 ((parse_number), gsmop5, set_gsmatch_min), !Minor id
: 455 0568 1 ((errorparse_1)));
: 456 P 0569 1 $state (gsmop5,
: 457 0570 1 ((endline_1), tpa$_exit));
: 458 0571 1
: 459 0572 1 Parse GSMATCH match control
: 460 0573 1
: 461 P 0574 1 $state (gsmatch_parse_control,
: 462 0575 1 ('ALWAYS', tpa$_exit, set_gsmatch_ctl, , , isd$k_matall),
: 463 P 0576 1 ('EQUAL', tpa$_exit, set_gsmatch_ctl, , , isd$k_matequ),
: 464 0577 1 ('LEQUAL', tpa$_exit, set_gsmatch_ctl, , , isd$k_matleq), ((errorparse_1)));
: 465 0578 1
: 466 0579 1 Parse IMAGE_TYPE option
: 467 0580 1
: 468 P 0581 1 $state (image_type_option,
: 469 0582 1 ('IMAGE_TYPE'));
: 470 P 0583 1 $state (
: 471 0584 1 ('='));
: 472 P 0585 1 $state (
: 473 0586 1 ((image_type_parse), imageop1), ((errorparse_1)));
: 474 P 0587 1 $state (imageop1,
: 475 0588 1 ((endline_1), tpa$_exit));
: 476 P 0589 1 $state (image_type_parse,
: 477 0590 1 ('CLI', tpa$_exit, set_image_type), ((errorparse_1)));
: 478 0591 1
: 479 0592 1 Parse IDENT option
: 480 0593 1
: 481 P 0594 1 $state (image_ident_option,
: 482 0595 1 ('IDENTIFICATION'));
: 483 P 0596 1 $state (
: 484 0597 1 ('='));

```

```

485 P 0598 1 $state (
486 P 0599 1 {'' tpa$_exit, set_image_ident, delimited),
487 0600 1 (tpa$_symbol, tpa$_exit, set_image_ident, , , not delimited), ((errorparse_1)));
488 0601 1
489 0602 1 Parse NAME option
490 0603 1
491 P 0604 1 $state (image_name_option,
492 0605 1 ('NAME'));
493 P 0606 1 $state (
494 0607 1 ('='));
495 P 0608 1 $state (
496 0609 1 ((get_symbol), nameop1, set_image_name), ((errorparse_1)));
497 P 0610 1 $state (nameop1,
498 0611 1 ((endline_1), tpa$_exit));
499 0612 1
500 0613 1 Parse IOSEGMENT option
501 0614 1
502 P 0615 1 $state (iosegment_option,
503 0616 1 ('IOSEGMENT'));
504 P 0617 1 $state (
505 0618 1 ('='));
506 P 0619 1 $state (
507 0620 1 ((parse_number), iosegop1, set_ioseg), !Number of buffers
508 0621 1 ((errorparse_1)));
509 P 0622 1 $state (iosegop1,
510 0623 1 (tpa$eos, tpa$_exit), !POBUFS keyword is optional
511 0624 1 ('', iosegop2), ((errorparse_1)));
512 P 0625 1 $state (iosegop2,
513 0626 1 ('POBUFS', tpa$_exit, set_p0bufs, , , 0),
514 P 0627 1 ('NOPOBUFS', tpa$_exit, set_p0bufs, , , 1),
515 0628 1 ((errorparse_1)));
516 0629 1
517 0630 1 Parse ISD_MAX option
518 0631 1
519 P 0632 1 $state (isd_max_option,
520 0633 1 ('ISD_MAX'));
521 P 0634 1 $state (
522 0635 1 ('='));
523 P 0636 1 $state (
524 0637 1 ((parse_number), isdmaxop1, set_isd_max), ((errorparse_1)));
525 P 0638 1 $state (isdmaxop1,
526 0639 1 ((endline_1), tpa$_exit));
527 0640 1
528 0641 1 Parse PROTECT option
529 0642 1
530 P 0643 1 $state (protect_option,
531 0644 1 ('PROTECT'));
532 P 0645 1 $state (
533 0646 1 ('='));
534 P 0647 1 $state (
535 0648 1 ('YES', protop1, set_protect, , , 1),
536 0649 1 ('NO', protop1, set_protect, , , 0), ((errorparse_1)));
537 P 0650 1 $state (protop1,
538 0651 1 ((endline_1), tpa$_exit));
539 0652 1
540 0653 1 Parse PSECT option
541 0654 1

```



```

542 P 0655 1 $state (psect_option,
543 0656 1 ('PSECT_ATTRIBUTES'));
544 P 0657 1 $state (
545 0658 1 {'='});
546 P 0659 1 $state (
547 0660 1 ((get_symbol), pscop1, createpsect), !Psect name
548 0661 1 ((errorparse_1)));
549 P 0662 1 $state (pscop1,
550 0663 1 ('', pscop2), ((errorparse_1)));
551 P 0664 1 $state (pscop2,
552 0665 1 ((psect_parse_attribute), pscop3)); !Followed by a list of attributes
553 P 0666 1 $state (pscop3,
554 0667 1 ('', pscop2), ! separated by commas
555 0668 1 (tpa$eos, tpa$exit), !Quit at end of line
556 0669 1 ((errorparse_1)));
557 0670 1
558 0671 1 Parse PSECT option attributes
559 0672 1
560 P 0673 1 $state (psect_parse_attribute,
561 0674 1 ('ABS', tpa$exit, set_pscatrib, , , ((not gps$m_rel) and rh_mask)),
562 0675 1 ('REL', tpa$exit, set_pscatrib, , , gps$m_rel),
563 0676 1 ('CON', tpa$exit, set_pscatrib, , , ((not gps$m_ovr) and rh_mask)),
564 0677 1 ('OVR', tpa$exit, set_pscatrib, , , gps$m_ovr),
565 0678 1 ('EXE', tpa$exit, set_pscatrib, , , (gps$m_exe or gps$m_rd)),
566 0679 1 ('NOEXE', tpa$exit, set_pscatrib, , , ((not gps$m_exe) and rh_mask)),
567 0680 1 ('GBL', tpa$exit, set_pscatrib, , , gps$m_gbl),
568 0681 1 ('LCL', tpa$exit, set_pscatrib, , , ((not gps$m_gbl) and rh_mask)),
569 0682 1 ('LIB', tpa$exit, set_pscatrib, , , gps$m_lib),
570 0683 1 ('USR', tpa$exit, set_pscatrib, , , ((not gps$m_lib) and rh_mask)),
571 0684 1 ('PIC', tpa$exit, set_pscatrib, , , gps$m_pic),
572 0685 1 ('NOPIC', tpa$exit, set_pscatrib, , , ((not gps$m_pic) and rh_mask)),
573 0686 1 ('RD', tpa$exit, set_pscatrib, , , gps$m_rd),
574 0687 1 ('NORD', tpa$exit, set_pscatrib, , , ((not gps$m_rd) and rh_mask)),
575 0688 1 ('SHR', tpa$exit, set_pscatrib, , , gps$m_shr),
576 0689 1 ('NOSHR', tpa$exit, set_pscatrib, , , ((not gps$m_shr) and rh_mask)),
577 0690 1 ('WRT', tpa$exit, set_pscatrib, , , (gps$m_rd or gps$m_wrt)),
578 0691 1 ('NOWRT', tpa$exit, set_pscatrib, , , ((not gps$m_wrt) and rh_mask)),
579 0692 1 ('VEC', tpa$exit, set_pscatrib, , , gps$m_vec),
580 0693 1 ('NOVEC', tpa$exit, set_pscatrib, , , ((not gps$m_vec) and rh_mask)),
581 0694 1 ('BYTE', tpa$exit, set_pscatrib, , , (0 or sign_bit)),
582 0695 1 ('WORD', tpa$exit, set_pscatrib, , , (1 or sign_bit)),
583 0696 1 ('LONG', tpa$exit, set_pscatrib, , , (2 or sign_bit)),
584 0697 1 ('QUAD', tpa$exit, set_pscatrib, , , (3 or sign_bit)),
585 0698 1 ('PAGE', tpa$exit, set_pscatrib, , , (9 or sign_bit)),
586 0699 1
587 P 0700 1 Allow alignment to be specified numerically also
588 0701 1
589 P 0702 1 ('0', tpa$exit, set_pscatrib, , , (0 or sign_bit)),
590 0703 1 ('1', tpa$exit, set_pscatrib, , , (1 or sign_bit)),
591 0704 1 ('2', tpa$exit, set_pscatrib, , , (2 or sign_bit)),
592 P 0705 1 ('3', tpa$exit, set_pscatrib, , , (3 or sign_bit)),
593 0706 1 ('9', tpa$exit, set_pscatrib, , , (9 or sign_bit)), ((errorparse_1)));
594 0707 1
595 0708 1 Parse SHL_EXTRA option
596 0709 1
597 P 0710 1 $state (shl_option,
598 0711 1 ('SHL_EXTRA'));

```

```

: 599 P 0712 1 $state (,
: 600 0713 1 ('='));
: 601 P 0714 1 $state (,
: 602 0715 1 ((parse_number), shl1, set_shl), ((errorparse_1)));
: 603 P 0716 1 $state (shl1,
: 604 0717 1 ((endline_1), tpa$_exit));
: 605 0718 1
: 606 0719 1 Parse STACK option
: 607 0720 1
: 608 P 0721 1 $state (stack_option,
: 609 0722 1 ('STACK'));
: 610 P 0723 1 $state (,
: 611 0724 1 ('='));
: 612 P 0725 1 $state (,
: 613 0726 1 ((parse_number), stackop1, set_stack), ((errorparse_1)));
: 614 P 0727 1 $state (stackop1,
: 615 0728 1 ((endline_1), tpa$_exit));
: 616 0729 1
: 617 0730 1 Parse SYMBOL option
: 618 0731 1
: 619 P 0732 1 $state (symbol_option,
: 620 0733 1 ('SYMBOL'));
: 621 P 0734 1 $state (,
: 622 0735 1 ('='));
: 623 P 0736 1 $state (,
: 624 0737 1 ((get_symbol), symbolop1, definesymbolname), !Symbol name
: 625 0738 1 ((errorparse_1)));
: 626 P 0739 1 $state (symbolop1,
: 627 0740 1 ('', symbolop2), ((errorparse_1)));
: 628 P 0741 1 $state (symbolop2,
: 629 0742 1 ((parse_number), symbolop3, definesymbolval), ! and symbol value
: 630 0743 1 ((errorparse_1)));
: 631 P 0744 1 $state (symbolop3,
: 632 0745 1 ((endline_1), tpa$_exit));
: 633 0746 1
: 634 0747 1 Parse UNSUPPORTED option
: 635 0748 1
: 636 P 0749 1 $state (unsupported_option,
: 637 0750 1 ('UNSUPPORTED'));
: 638 P 0751 1 $state (,
: 639 0752 1 ('=', unsupport1));
: 640 P 0753 1 $state (unsupport1,
: 641 0754 1 ((parse_number), unsupport2, set_unsupported), ((errorparse_1)));
: 642 P 0755 1 $state (unsupport2,
: 643 0756 1 ((endline_1), tpa$_exit));
: 644 0757 1
: 645 0758 1 Parse UNIVERSAL option
: 646 0759 1
: 647 P 0760 1 $state (universal_option,
: 648 0761 1 ('UNIVERSAL'));
: 649 P 0762 1 $state (,
: 650 0763 1 ('='));
: 651 P 0764 1 $state (,
: 652 0765 1 ((parse_univ_symbols), tpa$_exit); !List of symbols to universalize
: 653 0766 1
: 654 0767 1 Parse the symbols in the UNIVERSAL option
: 655 0768 1

```



```
: 656 P 0769 1 $state (parse_univ_symbols,  
: 657 P 0770 1 ('*', prsuniv2),  
: 658 0771 1 ((get_symbol), prsuniv1, set_universal), ((errorparse_1)));  
: 659 P 0772 1 $state (prsuniv1,  
: 660 P 0773 1 ('.', parse_univ_symbols),  
: 661 P 0774 1 (tpa$eos, tpa$exit), !Quit at end of string  
: 662 0775 1 ((errorparse_1)));  
: 663 P 0776 1 $state (prsuniv2,  
: 664 P 0777 1 (tpa$eos, prsuniv3),  
: 665 0778 1 ((errorparse_1)));  
: 666 P 0779 1 $state (prsuniv3,  
: 667 0780 1 (tpa$lambda, tpa$exit, set_alluniv)); ! Flag all universal temporarily  
: 668 0781 1 !  
: 669 0782 1 ! Parse a comma-separated file list  
: 670 0783 1 !  
: 671 P 0784 1 $state (parse_filelist,  
: 672 0785 1 ((filespec), parsefl1, , , filespec_desc)); !Do a file specification  
: 673 P 0786 1 $state (parsefl1,  
: 674 P 0787 1 ('_', tpa$fail), !Return failure if underscore, that is part of option  
: 675 P 0788 1 ! syntax.  
: 676 P 0789 1 ('=', tpa$fail), !Return failure if it looks like an option  
: 677 P 0790 1 ('.', parse_filelist, processfile), !Process comma-separated list  
: 678 P 0791 1 ('/', parsefl2), !Process qualifiers  
: 679 P 0792 1 (tpa$eos, tpa$exit, processfile), !Quit at end  
: 680 0793 1 ((errorparse_1)));  
: 681 P 0794 1 $state (parsefl2,  
: 682 P 0795 1 ((include_qual), parsefl3, set_include, , , (fdb$m_shr or fdb$m_option or fdb$m_selsr)),  
: 683 P 0796 1 ('LIBRARY', parsefl3, set_library, , , (fdb$m_shr or fdb$m_option or fdb$m_selsr)),  
: 684 P 0797 1 ('SELECTIVE SEARCH', parsefl3, set_selective, , , (fdb$m_option or fdb$m_libr)),  
: 685 0798 1 ((share_qual), parsefl3, set_shareable, , , (fdb$m_libr)), ((errorparse_1)));  
: 686 P 0799 1 $state (parsefl3,  
: 687 P 0800 1 ('/', parsefl2), !Process all qualifiers  
: 688 0801 1 (tpa$lambda, parsefl4));  
: 689 P 0802 1 $state (parsefl4,  
: 690 P 0803 1 ('.', parse_filelist, processfile), !Comma means more files  
: 691 P 0804 1 (tpa$eos, tpa$exit, processfile), !Exit at end of string  
: 692 0805 1 ((errorparse_1)));  
: 693 0806 1 !  
: 694 0807 1 ! Detect and check general file specification  
: 695 0808 1 !  
: 696 P 0809 1 $state (filespec,  
: 697 0810 1 ((nodespec), fil0), (tpa$lambda, fil0)); ! General file spec check  
: 698 P 0811 1 $state (fil0,  
: 699 P 0812 1 ((devname), fil1),  
: 700 0813 1 (tpa$lambda, fil1));  
: 701 P 0814 1 $state (fil1,  
: 702 P 0815 1 ((direct), fil2),  
: 703 0816 1 (tpa$lambda, fil2));  
: 704 P 0817 1 $state (fil2,  
: 705 0818 1 (tpa$symbol));  
: 706 P 0819 1 $state (  
: 707 P 0820 1 ((type), fil3), ! Check for type and version  
: 708 0821 1 (tpa$lambda, fil3));  
: 709 P 0822 1 $state (fil3,  
: 710 P 0823 1 ((version), tpa$exit),  
: 711 0824 1 (tpa$lambda, tpa$exit));  
: 712 0825 1 !
```



```
713      0826 1  ! Recognize node name (with optional access string)
714      0827 1  !
715      P 0828 1  $state (nodespec,
716      0829 1  ((node1), node0));
717      P 0830 1  $state (node0,
718      0831 1  ((node1), node0),
719      0832 1  (tpa$_lambda, tpa$_exit));
720      P 0833 1  $state (node1,
721      0834 1  (tpa$_symbol));
722      P 0835 1  $state (
723      0836 1  (:', node3), (')));
724      P 0837 1  $state (node2,
725      0838 1  (:', node4),
726      0839 1  (tpa$_eos, tpa$_fail),
727      0840 1  (:', tpa$_fail),
728      0841 1  (tpa$_any, node2));
729      P 0842 1  $state (node3,
730      0843 1  (:', tpa$_exit));
731      P 0844 1  $state (node4,
732      0845 1  (:', node3));
733      0846 1  !
734      0847 1  ! Recognize device name
735      0848 1  !
736      P 0849 1  $state (devname,
737      0850 1  (tpa$_symbol));
738      P 0851 1  $state (
739      0852 1  (:', tpa$_exit));
740      0853 1  !
741      0854 1  ! Recognize directory
742      0855 1  !
743      P 0856 1  $state (direct,
744      0857 1  ('<'),
745      0858 1  ('['));
746      P 0859 1  $state (
747      0860 1  ((ufd), dir2),
748      0861 1  ((dirsub)));
749      P 0862 1  $state (dir2,
750      0863 1  ('>', tpa$_exit),
751      0864 1  (']', tpa$_exit));
752      0865 1  !
753      0866 1  ! Recognize string directories and subdirectories
754      0867 1  !
755      P 0868 1  $state (dirsub,
756      0869 1  (tpa$_symbol, dirsub1),
757      0870 1  ('-', dirsub1),
758      0871 1  (tpa$_lambda));
759      P 0872 1  $state (dirsub1,
760      0873 1  (:', dirsub),
761      0874 1  (tpa$_lambda, tpa$_exit));
762      0875 1  !
763      0876 1  ! Recognize ufd directory format
764      0877 1  !
765      P 0878 1  $state (ufd,
766      0879 1  (tpa$_octal));
767      P 0880 1  $state (
768      0881 1  (:', '));
769      P 0882 1  $state (
```

```
! Get the first node spec
! Then try to get more
! try for one
! but if we don't get one, the exit successfully with the one
```

```
! Check for access control
```

```
! Look for end of access control
```

```
! Second colon delimiting node spec
```

```
! Must terminate the node spec.
```

```
! Check for uic directory name
```

```
! Allow [-]
! Allow [.sub.sub]
```

```
! Loop for another directory name
```



```
770      0883 1      (tpa$_octal, tpa$_exit));
771      0884 1
772      0885 1      Recognize file type
773      0886 1
774      P 0887 1 $state (type,
775      0888 1      ('.'));
776      P 0889 1 $state (
777      P 0890 1      (tpa$_string, tpa$_exit),
778      P 0891 1      (tpa$_blank, tpa$_exit),
779      0892 1      (tpa$_lambda, tpa$_exit));
780      0893 1
781      0894 1      Recognize file version
782      0895 1
783      P 0896 1 $state (version,
784      0897 1      (';'));
785      P 0898 1 $state (
786      P 0899 1      (tpa$_decimal, tpa$_exit),
787      P 0900 1      (tpa$_blank, tpa$_exit),
788      0901 1      (tpa$_lambda, tpa$_exit));
789      0902 1
790      0903 1      Parse a number or a null field
791      0904 1
792      P 0905 1 $state (parse_number_or_null,
793      P 0906 1      ((parse_number), tpa$_exit),      !All ok if number
794      0907 1      (tpa$_lambda, tpa$_exit));      ! or else nothing
795      0908 1
796      0909 1      Recognize number
797      0910 1
798      P 0911 1 $state (parse_number,
799      P 0912 1      (tpa$_decimal, tpa$_exit),
800      0913 1      ('x'));
801      P 0914 1 $state (
802      P 0915 1      ('D', decnum),
803      P 0916 1      ('X', hexnum),
804      0917 1      ('O'));
805      P 0918 1 $state (
806      0919 1      (tpa$_octal, tpa$_exit));
807      P 0920 1 $state (hexnum,
808      0921 1      (tpa$_hex, tpa$_exit));
809      P 0922 1 $state (decnum,
810      0923 1      (tpa$_decimal, tpa$_exit));
811      0924 1
812      0925 1      Parse INCLUDE qualifier
813      0926 1
814      P 0927 1 $state (include_qual,
815      0928 1      ('INCLUDE'));      !Include
816      P 0929 1 $state (
817      P 0930 1      ((qualvalsep), incl1),      !Followed by '=' or ':'
818      0931 1      ((errorparse_1)));      ! or its an error
819      P 0932 1 $state (incl1,
820      P 0933 1      (('(', include_list),      !If a left parens
821      P 0934 1      ((get_symbol), tpa$_exit, ., include_desc),      ! or a symbol then ok
822      0935 1      ((errorparse_1)));      ! otherwise an error
823      0936 1
824      0937 1      Parse a list of symbols enclosed in parens
825      0938 1
826      P 0939 1 $state (include_list,
```

```
827      0940 1      ((parse_include_list), incl2, , , include_desc));
828      0941 1
829      P 0942 1 $state (incl2,
830      P 0943 1      ('', tpa$_exit),
831      0944 1      ((errorparse_1)));
832      P 0945 1 $state (parse_include_list,
833      P 0946 1      ((get_symbol), incl3, !Get one symbol
834      0947 1      ((errorparse_1)));
835      P 0948 1 $state (incl3,
836      P 0949 1      ('', parse_include_list),
837      0950 1      (tpa$_lambda, tpa$_exit));
838      0951 1
839      0952 1 Parse /SHARE qualifier
840      0953 1
841      P 0954 1 $state (share_qual,
842      0955 1      ('SHAREABLE', , set_shrcopyflag, , , 0));
843      P 0956 1 $state (
844      P 0957 1      ((qualvalsep), parseshareval),
845      0958 1      (tpa$_lambda, tpa$_exit));
846      P 0959 1 $state (parseshareval,
847      P 0960 1      ('COPY', tpa$_exit, set_shrcopyflag, , , 1),
848      0961 1      ('NOCOPY', tpa$_exit, set_shrcopyflag, , , 0), ((errorparse_1)));
849      0962 1
850      0963 1 Parse a symbol, which may include periods (.MAIN.)
851      0964 1
852      P 0965 1 $state (get_symbol,
853      P 0966 1      ('', getsym1),
854      P 0967 1      (tpa$_symbol, getsym1),
855      0968 1      (tpa$_lambda, tpa$_fail));
856      P 0969 1 $state (getsym1,
857      P 0970 1      ('', getsym1),
858      P 0971 1      (tpa$_symbol, getsym1),
859      0972 1      (tpa$_lambda, tpa$_exit));
860      0973 1
861      0974 1 Check qualifier and value separator
862      0975 1
863      P 0976 1 $state (qualvalsep,
864      P 0977 1      (':', tpa$_exit),
865      0978 1      ('=', tpa$_exit));
866      0979 1
867      0980 1 Try to parse something and then call the error routine
868      0981 1
869      P 0982 1 $state (errorparse_1,
870      P 0983 1      (tpa$_eos, missingargerr),
871      P 0984 1      (tpa$_symbol, , syntaxerr),
872      P 0985 1      (tpa$_string, , syntaxerr),
873      P 0986 1      (tpa$_any, , syntaxerr),
874      0987 1      (tpa$_lambda, , syntaxerr));
875      0988 1
876      0989 1 Check end of line conditions
877      0990 1
878      P 0991 1 $state (endline_1,
879      P 0992 1      (tpa$_eos, tpa$_exit),
880      P 0993 1      (tpa$_symbol, , syntaxerr),
881      P 0994 1      (tpa$_string, , syntaxerr),
882      0995 1      (tpa$_any, , syntaxerr));
```

```
!Call the routine and store the descriptor
!Followed by a right parens
! or a syntax error
!Get one symbol
! or an error
!Go for more if a comma
!otherwise return
!Zero flag
!Fail if not at least one character
!End of string is a good exit
!Else try to get a symbol
!Else try to get a string
! or anything to flag as error
```



```

: 884      0996 1 %sbttl 'readoption - Read record from options file'
: 885      0997 1 :
: 886      0998 1 routine readoption (linedesc, errorsignal) =
: 887      0999 1 :
: 888      1000 1   This routine reads and returns to the caller in linedesc the
: 889      1001 1   next record of the options file.  If end of file is detected, then
: 890      1002 1   one of two things can happen.  If the errorsignal argument is
: 891      1003 1   supplied, then the error is signaled.  If it is not supplied,
: 892      1004 1   rms$_eof is returned to the caller.
: 893      1005 1 :
: 894      1006 2 begin
: 895      1007 2 map
: 896      1008 2     linedesc      : ref bblock;
: 897      1009 2 builtin
: 898      1010 2     nullparameter;
: 899      1011 2 local
: 900      1012 2     status,
: 901      1013 2     blocksize,
: 902      1014 2     blockaddr      : ref bblock;
: 903      1015 2
: 904      1016 2 status = $get (rab = .optrabadr);           !Read the record
: 905      1017 2 linedesc [dsc$_length] = .optrabadr [rab$_rsz]; !Return it to caller
: 906      1018 2 linedesc [dsc$_pointer] = .optrabadr [rab$_rbf];
: 907      1019 2
: 908      1020 2 if .status                                !If a successful read
: 909      1021 2 then begin
: 910      1022 2     if .lnk$_ctlmsk [lnk$_verify]           !Echo line if SET VERIFY
: 911      1023 2     and not .lnk$_sysinput_flag             !and not SYSS$INPUT
: 912      1024 2     then lib$put_output (.linedesc);
: 913      1025 2
: 914      1026 2     blocksize = oeb$_size + .linedesc [dsc$_length]; !Figure size of block to allocate
: 915      1027 2     lnk$alloblk (.blocksize, blockaddr);      !Allocate it
: 916      1028 2     blockaddr [oeb$_l_nxttoeb] = 0;
: 917      1029 2     blockaddr [oeb$_bytcnt] = .linedesc [dsc$_length]; !Set size of line into descriptor
: 918      1030 2     ch$move (.linedesc [dsc$_length]         ! and copy out the text
: 919      1031 2     , .linedesc [dsc$_pointer]
: 920      1032 2     , blockaddr [oeb$_text]);
: 921      1033 2     lnk$_optexte [oeb$_l_nxttoeb] = .blockaddr; !Link the block into the end of the list
: 922      1034 2     lnk$_optexte = .blockaddr;               ! and make it the last one
: 923      1035 2 end
: 924      1036 2 else begin
: 925      1037 2 :
: 926      1038 2     An error occurred on the read
: 927      1039 2 :
: 928      1040 2     if .status eql rms$_eof                !If this is end of file
: 929      1041 2     then begin
: 930      1042 2         if not nullparameter (2)             ! and signal code supplied
: 931      1043 2         then signal (.errorsignal, 1, .optionfilename ! then signal it (eof not expected)
: 932      1044 2         , status, .optrabadr [rab$_stx]
: 933      1045 2         );
: 934      1046 2     end
: 935      1047 2     else signal_stop (lin$_readerr, 1, .optionfilename !Not eof, signal the read error and give up
: 936      1048 2     , status, .optrabadr [rab$_stx]
: 937      1049 2     );
: 938      1050 2 end;
: 939      1051 2
: 940      1052 2 return .status
```

LNK_PROCOPTIONS Linker options parser
V04=000 readoption - Read record from options file
; 941 1053 1 end;

F 13
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 18
(3)

```
.TITLE LNK_PROCOPTIONS Linker options parser
.IDENT \V04-000\

.PSECT _LIB$KEY1$,NOWRT, SHR, PIC,1

00000 ;TPASKEYSTO
      U.78: .BLKB 0
      45 53 41 42 00000 ;TPASKEYST
      U.80: .ASCII \BASE\
      FF 00004 .BYTE -1
      FF 00005 ;TPASKEYFILL
      U.82: .BYTE -1
      00006 ;TPASKEYSTO
      U.99: .BLKB 0
      53 4C 45 4E 4E 41 48 43 00006 ;TPASKEYST
      U.101: .ASCII \CHANNELS\
      FF 0000E .BYTE -1
      FF 0000F ;TPASKEYFILL
      U.103: .BYTE -1
      00010 ;TPASKEYSTO
      U.117: .BLKB 0
      52 45 54 53 55 4C 43 00010 ;TPASKEYST
      U.119: .ASCII \CLUSTER\
      FF 00017 .BYTE -1
      FF 00018 ;TPASKEYFILL
      U.121: .BYTE -1
      00019 ;TPASKEYSTO
      U.190: .BLKB 0
      54 43 45 4C 4C 4F 43 00019 ;TPASKEYST
      U.192: .ASCII \COLLECT\
      FF 00020 .BYTE -1
      FF 00021 ;TPASKEYFILL
      U.194: .BYTE -1
      00022 ;TPASKEYSTO
      U.231: .BLKB 0
      4E 49 4D 5F 4F 52 5A 44 00022 ;TPASKEYST
      U.233: .ASCII \DZRO_MIN\
      FF 0002A .BYTE -1
      FF 0002B ;TPASKEYFILL
      U.235: .BYTE -1
      0002C ;TPASKEYSTO
      U.249: .BLKB 0
      48 43 54 41 4D 53 47 0002C ;TPASKEYST
      U.251: .ASCII \GSMATCH\
      FF 00033 .BYTE -1
      FF 00034 ;TPASKEYFILL
      U.253: .BYTE -1
      00035 ;TPASKEYSTO
      U.289: .BLKB 0
      53 59 41 57 4C 41 00035 ;TPASKEYST
      U.291: .ASCII \ALWAYS\
      FF 0003B .BYTE -1
      0003C ;TPASKEYSTO
      U.297: .BLKB 0
```


Offset	Hex	ASCII	Comment
4C 41 55 51 45	0003C	;TPASKEYST	
	FF 00041	U.299: .ASCII	\EQUAL\
	00042	;TPASKEYSTO	
4C 41 55 51 45 4C	00042	U.305: .BLKB	0
	FF 00048	U.307: .ASCII	\LEQUAL\
	FF 00049	;TPASKEYFILL	
	0004A	U.315: .BYTE	-1
	0004A	;TPASKEYSTO	
45 50 59 54 5F 45 47 41 4D 49	0004A	U.316: .BLKB	0
	FF 00054	U.318: .ASCII	\IMAGE_TYPE\
	FF 00055	;TPASKEYFILL	
	00056	U.320: .BYTE	-1
	00056	;TPASKEYSTO	
	00056	U.332: .BLKB	0
49 4C 43	00056	;TPASKEYST	
	FF 00059	U.334: .ASCII	\CLI\
	FF 0005A	;TPASKEYFILL	
	0005B	U.340: .BYTE	-1
	0005B	;TPASKEYSTO	
4E 4F 49 54 41 43 49 46 49 54 4E 45 44 49	0005B	U.341: .BLKB	0
	FF 00069	U.343: .ASCII	\IDENTIFICATION\
	FF 0006A	;TPASKEYFILL	
	0006B	U.345: .BYTE	-1
	0006B	;TPASKEYSTO	
	0006B	U.359: .BLKB	0
45 4D 41 4E	0006B	;TPASKEYST	
	FF 0006F	U.361: .ASCII	\NAME\
	FF 00070	;TPASKEYFILL	
	00071	U.363: .BYTE	-1
	00071	;TPASKEYSTO	
	00071	U.375: .BLKB	0
54 4E 45 4D 47 45 53 4F 49	00071	;TPASKEYST	
	FF 0007A	U.377: .ASCII	\IOSEGMENT\
	FF 0007B	;TPASKEYFILL	
	0007C	U.379: .BYTE	-1
	0007C	;TPASKEYSTO	
	0007C	U.395: .BLKB	0
53 46 55 42 30 50	0007C	;TPASKEYST	
	FF 00082	U.397: .ASCII	\POBUFS\
	00083	;TPASKEYSTO	
	00083	U.403: .BLKB	0
53 46 55 42 30 50 4F 4E	00083	;TPASKEYST	
	FF 0008B	U.405: .ASCII	\NOPOBUFS\
	FF 0008C	;TPASKEYFILL	
		U.413: .BYTE	-1

[illegible]


```

      FF 000CD U.522: .ASCII \EXE\
      000CE :TPASKEYSTO .BYTE -1
45 58 45 4F 4E 000CE U.528: .BLKB 0
      :TPASKEYST
      FF 000D3 U.530: .ASCII \NOEXE\
      000D4 :TPASKEYSTO .BYTE -1
      U.536: .BLKB 0
4C 42 47 000D4 :TPASKEYST
      U.538: .ASCII \GBL\
      FF 000D7 :TPASKEYSTO .BYTE -1
      000D8 :TPASKEYSTO
      U.544: .BLKB 0
4C 43 4C 000D8 :TPASKEYST
      U.546: .ASCII \LCL\
      FF 000DB :TPASKEYSTO .BYTE -1
      000DC :TPASKEYSTO
      U.552: .BLKB 0
42 49 4C 000DC :TPASKEYST
      U.554: .ASCII \LIB\
      FF 000DF :TPASKEYSTO .BYTE -1
      000E0 :TPASKEYSTO
      U.560: .BLKB 0
52 53 55 000E0 :TPASKEYST
      U.562: .ASCII \USR\
      FF 000E3 :TPASKEYSTO .BYTE -1
      000E4 :TPASKEYSTO
      U.568: .BLKB 0
43 49 50 000E4 :TPASKEYST
      U.570: .ASCII \PIC\
      FF 000E7 :TPASKEYSTO .BYTE -1
      000E8 :TPASKEYSTO
      U.576: .BLKB 0
43 49 50 4F 4E 000E8 :TPASKEYST
      U.578: .ASCII \NOPIC\
      FF 000ED :TPASKEYSTO .BYTE -1
      000EE :TPASKEYSTO
      U.584: .BLKB 0
      44 52 000EE :TPASKEYST
      U.586: .ASCII \RD\
      FF 000F0 :TPASKEYSTO .BYTE -1
      000F1 :TPASKEYSTO
      U.592: .BLKB 0
44 52 4F 4E 000F1 :TPASKEYST
      U.594: .ASCII \NORD\
      FF 000F5 :TPASKEYSTO .BYTE -1
      000F6 :TPASKEYSTO
      U.600: .BLKB 0
      52 48 53 000F6 :TPASKEYST
      U.602: .ASCII \SHR\
      FF 000F9 :TPASKEYSTO .BYTE -1
      000FA :TPASKEYSTO
      U.608: .BLKB 0
52 48 53 4F 4E 000FA :TPASKEYST
      U.610: .ASCII \NOSHR\
      FF 000FF :TPASKEYSTO .BYTE -1

```

				00100	;TPASKEYSTO	
					U.616: .BLKB	0
54	52	57		00100	;TPASKEYST	
					U.618: .ASCII	\WRT\
			FF	00103	.BYTE	-1
				00104	;TPASKEYSTO	
					U.624: .BLKB	0
54	52	57	4F	4E	00104	;TPASKEYST
					U.626: .ASCII	\NOWRT\
			FF	00109	.BYTE	-1
				0010A	;TPASKEYSTO	
					U.632: .BLKB	0
43	45	56		0010A	;TPASKEYST	
					U.634: .ASCII	\VEC\
			FF	0010D	.BYTE	-1
				0010E	;TPASKEYSTO	
					U.640: .BLKB	0
43	45	56	4F	4E	0010E	;TPASKEYST
					U.642: .ASCII	\NOVEC\
			FF	00113	.BYTE	-1
				00114	;TPASKEYSTO	
					U.648: .BLKB	0
45	54	59	42	00114	;TPASKEYST	
					U.650: .ASCII	\BYTE\
			FF	00118	.BYTE	-1
				00119	;TPASKEYSTO	
					U.656: .BLKB	0
44	52	4F	57	00119	;TPASKEYST	
					U.658: .ASCII	\WORD\
			FF	0011D	.BYTE	-1
				0011E	;TPASKEYSTO	
					U.664: .BLKB	0
47	4E	4F	4C	0011E	;TPASKEYST	
					U.666: .ASCII	\LONG\
			FF	00122	.BYTE	-1
				00123	;TPASKEYSTO	
					U.672: .BLKB	0
44	41	55	51	00123	;TPASKEYST	
					U.674: .ASCII	\QUAD\
			FF	00127	.BYTE	-1
				00128	;TPASKEYSTO	
					U.680: .BLKB	0
45	47	41	50	00128	;TPASKEYST	
					U.682: .ASCII	\PAGE\
			FF	0012C	.BYTE	-1
			FF	0012D	;TPASKEYFILL	
					U.715: .BYTE	-1
				0012E	;TPASKEYSTO	
					U.716: .BLKB	0
41	52	54	58	45	5F	4C
				48	53	0012E
						;TPASKEYST
						U.718: .ASCII
			FF	00137	.BYTE	\SHL_EXTRA\
			FF	00138	.BYTE	-1
					;TPASKEYFILL	
					U.720: .BYTE	-1
				00139	;TPASKEYSTO	
					U.732: .BLKB	0
48	43	41	54	53	00139	;TPASKEYST

Offset	Hex	ASCII	Comment
0013E	FF	U.734: .ASCII	\STACK\
0013F	FF	;TPASKEYFILL	-1
00140		U.736: .BYTE	-1
00140		;TPASKEYSTO	0
00140	4C 4F 42 4D 59 53	U.748: .BLKB	0
00140		;TPASKEYST	
00146	FF	U.750: .ASCII	\SYMBOL\
00147	FF	;TPASKEYFILL	-1
00148		U.752: .BYTE	-1
00148		;TPASKEYSTO	0
00148	44 45 54 52 4F 50 50 55 53 4E 55	U.776: .BLKB	0
00148		;TPASKEYST	
00153		U.778: .ASCII	\UNSUPPORTED\
00154	FF	;TPASKEYFILL	-1
00155		U.780: .BYTE	-1
00155		;TPASKEYSTO	0
00155	4C 41 53 52 45 56 49 4E 55	U.794: .BLKB	0
00155		;TPASKEYST	
0015E	FF	U.796: .ASCII	\UNIVERSAL\
0015F	FF	;TPASKEYFILL	-1
00160		U.798: .BYTE	-1
00160		;TPASKEYSTO	0
00160	59 52 41 52 42 49 4C	U.857: .BLKB	0
00160		;TPASKEYST	
00167	FF	U.859: .ASCII	\LIBRARY\
00168		;TPASKEYSTO	-1
00168	43 52 41 45 53 5F 45 56 49 54 43 45 4C 45 53	U.865: .BLKB	0
00168		;TPASKEYST	
00168		U.867: .ASCII	\SELECTIVE_SEARCH\
00177	48		
00178	FF	;TPASKEYFILL	-1
00179	FF	U.882: .BYTE	-1
0017A		;TPASKEYSTO	0
0017A	45 44 55 4C 43 4E 49	U.1025: .BLKB	0
0017A		;TPASKEYST	
00181	FF	U.1027: .ASCII	\INCLUDE\
00182	FF	;TPASKEYFILL	-1
00183		U.1029: .BYTE	-1
00183		;TPASKEYSTO	0
00183	45 4C 42 41 45 52 41 48 53	U.1066: .BLKB	0
00183		;TPASKEYST	
00183		U.1068: .ASCII	\SHAREABLE\
0018C	FF	;TPASKEYFILL	-1
0018D	FF	U.1073: .BYTE	-1
0018E		;TPASKEYSTO	0
0018E	59 50 4F 43 4F 4E	U.1080: .BLKB	0
0018E		;TPASKEYST	
0018E		U.1082: .ASCII	\NOCOPY\
00194	FF	;TPASKEYFILL	-1

```

FF 00195 ;TPASKEYFILL
      U.1090: .BYTE -1
      .PSECT _LIB$STATES$,NOWRT, SHR, PIC,1
00000 STATE_TABLE::
      .BLKB 0
00000 OPTION: .BLKB 0
19F8 00000 ;TPASTYPE
      U.2: .WORD 6648
0000* 00002 ;TPASSUBEXP
      U.4: .WORD <<U.3-U.4>-2>
FFFF 00004 ;TPASTARGET
      U.5: .WORD -1
19F8 00006 ;TPASTYPE
      U.6: .WORD 6648
0000* 00008 ;TPASSUBEXP
      U.8: .WORD <<U.7-U.8>-2>
FFFF 0000A ;TPASTARGET
      U.9: .WORD -1
19F8 0000C ;TPASTYPE
      U.10: .WORD 6648
0000* 0000E ;TPASSUBEXP
      U.12: .WORD <<U.11-U.12>-2>
FFFF 00010 ;TPASTARGET
      U.13: .WORD -1
19F8 00012 ;TPASTYPE
      U.14: .WORD 6648
0000* 00014 ;TPASSUBEXP
      U.16: .WORD <<U.15-U.16>-2>
FFFF 00016 ;TPASTARGET
      U.17: .WORD -1
19F8 00018 ;TPASTYPE
      U.18: .WORD 6648
0000* 0001A ;TPASSUBEXP
      U.20: .WORD <<U.19-U.20>-2>
FFFF 0001C ;TPASTARGET
      U.21: .WORD -1
19F8 0001E ;TPASTYPE
      U.22: .WORD 6648
0000* 00020 ;TPASSUBEXP
      U.24: .WORD <<U.23-U.24>-2>
FFFF 00022 ;TPASTARGET
      U.25: .WORD -1
19F8 00024 ;TPASTYPE
      U.26: .WORD 6648
0000* 00026 ;TPASSUBEXP
      U.28: .WORD <<U.27-U.28>-2>
FFFF 00028 ;TPASTARGET
      U.29: .WORD -1
19F8 0002A ;TPASTYPE
      U.30: .WORD 6648
0000* 0002C ;TPASSUBEXP
      U.32: .WORD <<U.31-U.32>-2>
FFFF 0002E ;TPASTARGET
      U.33: .WORD -1
19F8 00030 ;TPASTYPE

```


0000*	00032	U.34: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	00034	U.36: .WORD	<<U.35-U.36>-2>	:
		:TPASTARGET		:
19F8	00036	U.37: .WORD	-1	:
		:TPASTYPE		:
0000*	00038	U.38: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	0003A	U.40: .WORD	<<U.39-U.40>-2>	:
		:TPASTARGET		:
19F8	0003C	U.41: .WORD	-1	:
		:TPASTYPE		:
0000*	0003E	U.42: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	00040	U.44: .WORD	<<U.43-U.44>-2>	:
		:TPASTARGET		:
19F8	00042	U.45: .WORD	-1	:
		:TPASTYPE		:
0000*	00044	U.46: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	00046	U.48: .WORD	<<U.47-U.48>-2>	:
		:TPASTARGET		:
19F8	00048	U.49: .WORD	-1	:
		:TPASTYPE		:
0000*	0004A	U.50: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	0004C	U.52: .WORD	<<U.51-U.52>-2>	:
		:TPASTARGET		:
19F8	0004E	U.53: .WORD	-1	:
		:TPASTYPE		:
0000*	00050	U.54: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	00052	U.56: .WORD	<<U.55-U.56>-2>	:
		:TPASTARGET		:
19F8	00054	U.57: .WORD	-1	:
		:TPASTYPE		:
0000*	00056	U.58: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	00058	U.60: .WORD	<<U.59-U.60>-2>	:
		:TPASTARGET		:
19F8	0005A	U.61: .WORD	-1	:
		:TPASTYPE		:
0000*	0005C	U.62: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	0005E	U.64: .WORD	<<U.63-U.64>-2>	:
		:TPASTARGET		:
19F8	00060	U.65: .WORD	-1	:
		:TPASTYPE		:
0000*	00062	U.66: .WORD	6648	:
		:TPASSUBEXP		:
FFFF	00064	U.68: .WORD	<<U.67-U.68>-2>	:
		:TPASTARGET		:
19F8	00066	U.69: .WORD	-1	:
		:TPASTYPE		:
0000*	00068	U.70: .WORD	6648	:
		:TPASSUBEXP		:
		U.72: .WORD	<<U.71-U.72>-2>	:

```

FFFF 0006A :TPASTARGET
          U.73: .WORD -1
1DF8 0006C :TPASTYPE
          U.74: .WORD 7672
0000* 0006E :TPASSUBEXP
          U.76: .WORD <<U.75-U.76>-2>
FFFF 00070 :TPASTARGET
          U.77: .WORD -1
          00072 :BASE_OPTION
          U.7: .BLKB 0
0500 00072 :TPASTYPE
          U.81: .WORD 1280
143D 00074 :TPASTYPE
          U.83: .WORD 5181
0000* 00076 :TPASTARGET
          U.85: .WORD <<U.84-U.85>-2>
          00078 :BASEOP1
          U.84: .BLKB 0
99F8 00078 :TPASTYPE
          U.86: .WORD -26120
0000* 0007A :TPASSUBEXP
          U.88: .WORD <<U.87-U.88>-2>
00000000V 0007C :TPASACTION
          U.89: .LONG <<SET_BASE-U.89>-4>
0000* 00080 :TPASTARGET
          U.91: .WORD <<U.90-U.91>-2>
0DF8 00082 :TPASTYPE
          U.92: .WORD 3576
0000* 00084 :TPASSUBEXP
          U.94: .WORD <<U.93-U.94>-2>
          00086 :BASEOP2
          U.90: .BLKB 0
1DF8 00086 :TPASTYPE
          U.95: .WORD 7672
0000* 00088 :TPASSUBEXP
          U.97: .WORD <<U.96-U.97>-2>
FFFF 0008A :TPASTARGET
          U.98: .WORD -1
          0008C :CHANNELS_OPTION
          U.11: .BLKB 0
0501 0008C :TPASTYPE
          U.102: .WORD 1281
143D 0008E :TPASTYPE
          U.104: .WORD 5181
0000* 00090 :TPASTARGET
          U.106: .WORD <<U.105-U.106>-2>
          00092 :CHANOP1
          U.105: .BLKB 0
99F8 00092 :TPASTYPE
          U.107: .WORD -26120
0000* 00094 :TPASSUBEXP
          U.108: .WORD <<U.87-U.108>-2>
00000000V 00096 :TPASACTION
          U.109: .LONG <<SET_CHANNELS-U.109>-4>
0000* 0009A :TPASTARGET
          U.111: .WORD <<U.110-U.111>-2>
0DF8 0009C :TPASTYPE

```


		U.112: .WORD	3576	:
0000*	0009E	:TPASSUBEXP		:
		U.113: .WORD	<<U.93-U.113>-2>	:
	000A0	:CHANOP2		:
		U.110: .BLKB	0	:
1DF8	000A0	:TPASTYPE		:
		U.114: .WORD	7672	:
0000*	000A2	:TPASSUBEXP		:
		U.115: .WORD	<<U.96-U.115>-2>	:
FFFF	000A4	:TPASTARGET		:
		U.116: .WORD	-1	:
	000A6	:CLUSTER_OPTION		:
		U.15: .BLKB	0	:
0502	000A6	:TPASTYPE		:
		U.120: .WORD	1282	:
043D	000A8	:TPASTYPE		:
		U.122: .WORD	1085	:
99F8	000AA	:TPASTYPE		:
		U.123: .WORD	-26120	:
0000*	000AC	:TPASSUBEXP		:
		U.125: .WORD	<<U.124-U.125>-2>	:
00000000V	000AE	:TPASACTION		:
		U.126: .LONG	<<CREATECLUSTER-U.126>-4>	:
0000*	000B2	:TPASTARGET		:
		U.128: .WORD	<<U.127-U.128>-2>	:
0DF8	000B4	:TPASTYPE		:
		U.129: .WORD	3576	:
0000*	000B6	:TPASSUBEXP		:
		U.130: .WORD	<<U.93-U.130>-2>	:
	000B8	:CLUOP1		:
		U.127: .BLKB	0	:
11F7	000B8	:TPASTYPE		:
		U.131: .WORD	4599	:
0000*	000BA	:TPASTARGET		:
		U.133: .WORD	<<U.132-U.133>-2>	:
102C	000BC	:TPASTYPE		:
		U.134: .WORD	4140	:
0000*	000BE	:TPASTARGET		:
		U.136: .WORD	<<U.135-U.136>-2>	:
0DF8	000C0	:TPASTYPE		:
		U.137: .WORD	3576	:
0000*	000C2	:TPASSUBEXP		:
		U.138: .WORD	<<U.93-U.138>-2>	:
	000C4	:CLUOP2		:
		U.135: .BLKB	0	:
11F7	000C4	:TPASTYPE		:
		U.139: .WORD	4599	:
0000*	000C6	:TPASTARGET		:
		U.140: .WORD	<<U.132-U.140>-2>	:
99F8	000C8	:TPASTYPE		:
		U.141: .WORD	-26120	:
0000*	000CA	:TPASSUBEXP		:
		U.143: .WORD	<<U.142-U.143>-2>	:
00000000V	000CC	:TPASACTION		:
		U.144: .LONG	<<SETCLUSTERBASE-U.144>-4>	:
0000*	000D0	:TPASTARGET		:
		U.146: .WORD	<<U.145-U.146>-2>	:

0DF8	000D2	:TPASTYPE			
		U.147: .WORD	3576		:
0000*	000D4	:TPASSUBEXP			
		U.148: .WORD	<<U.93-U.148>-2>		:
	000D6	:CLUOP3			
		U.145: .BLKB	0		
11F7	000D6	:TPASTYPE			
		U.149: .WORD	4599		:
0000*	000D8	:TPASTARGET			
		U.150: .WORD	<<U.132-U.150>-2>		:
102C	000DA	:TPASTYPE			
		U.151: .WORD	4140		:
0000*	000DC	:TPASTARGET			
		U.153: .WORD	<<U.152-U.153>-2>		:
0DF8	000DE	:TPASTYPE			
		U.154: .WORD	3576		:
0000*	000E0	:TPASSUBEXP			
		U.155: .WORD	<<U.93-U.155>-2>		:
	000E2	:CLUOP4			
		U.152: .BLKB	0		
11F7	000E2	:TPASTYPE			
		U.156: .WORD	4599		:
0000*	000E4	:TPASTARGET			
		U.157: .WORD	<<U.132-U.157>-2>		:
99F8	000E6	:TPASTYPE			
		U.158: .WORD	-26120		:
0000*	000E8	:TPASSUBEXP			
		U.159: .WORD	<<U.142-U.159>-2>		:
00000000V	000EA	:TPASACTION			
		U.160: .LONG	<<SETCLUSTERPFC-U.160>-4>		:
0000*	000EE	:TPASTARGET			
		U.162: .WORD	<<U.161-U.162>-2>		:
0DF8	000F0	:TPASTYPE			
		U.163: .WORD	3576		:
0000*	000F2	:TPASSUBEXP			
		U.164: .WORD	<<U.93-U.164>-2>		:
	000F4	:CLUOP5			
		U.161: .BLKB	0		
11F7	000F4	:TPASTYPE			
		U.165: .WORD	4599		:
0000*	000F6	:TPASTARGET			
		U.166: .WORD	<<U.132-U.166>-2>		:
102C	000F8	:TPASTYPE			
		U.167: .WORD	4140		:
0000*	000FA	:TPASTARGET			
		U.169: .WORD	<<U.168-U.169>-2>		:
0DF8	000FC	:TPASTYPE			
		U.170: .WORD	3576		:
0000*	000FE	:TPASSUBEXP			
		U.171: .WORD	<<U.93-U.171>-2>		:
	00100	:CLUOP6			
		U.168: .BLKB	0		
11F7	00100	:TPASTYPE			
		U.172: .WORD	4599		:
0000*	00102	:TPASTARGET			
		U.173: .WORD	<<U.132-U.173>-2>		:
19F8	00104	:TPASTYPE			


```

0000* 00106 U.174: .WORD 6648 ;
          :TPASSUBEXP ;
0000* 00108 U.175: .WORD <<U.3-U.175>-2> ;
          :TPASTARGET ;
0DF8 0010A U.177: .WORD <<U.176-U.177>-2> ;
          :TPASTYPE ;
0000* 0010C U.178: .WORD 3576 ;
          :TPASSUBEXP ;
          0010E U.179: .WORD <<U.93-U.179>-2> ;
          :CLUOP7 ;
1DF8 0010E U.176: .BLKB 0 ;
          :TPASTYPE ;
0000* 00110 U.180: .WORD 7672 ;
          :TPASSUBEXP ;
0000* 00112 U.181: .WORD <<U.96-U.181>-2> ;
          :TPASTARGET ;
          00114 U.183: .WORD <<U.182-U.183>-2> ;
          :CLUOP7A ;
95F6 00114 U.132: .BLKB 0 ;
          :TPASTYPE ;
00000000V 00116 U.184: .WORD -27146 ;
          :TPASACTION ;
0000* 0011A U.185: .LONG <<INSERTCLUSTER-U.185>-4> ;
          :TPASTARGET ;
          0011C U.186: .WORD <<U.182-U.186>-2> ;
          :CLUOP8 ;
95F6 0011C U.182: .BLKB 0 ;
          :TPASTYPE ;
00000000V 0011E U.187: .WORD -27146 ;
          :TPASACTION ;
          00122 U.188: .LONG <<CLUSTERDONE-U.188>-4> ;
          :TPASTARGET ;
          00124 U.189: .WORD -1 ;
          :COLLECT_OPTION ;
          00124 U.19: .BLKB 0 ;
          :TPASTYPE ;
0503 00124 U.193: .WORD 1283 ;
          :TPASTYPE ;
043D 00126 U.195: .WORD 1085 ;
          :TPASTYPE ;
99F8 00128 U.196: .WORD -26120 ;
          :TPASSUBEXP ;
0000* 0012A U.197: .WORD <<U.124-U.197>-2> ;
          :TPASACTION ;
00000000V 0012C U.198: .LONG <<SET_COLLECT-U.198>-4> ;
          :TPASTARGET ;
          00130 U.200: .WORD <<U.199-U.200>-2> ;
          :TPASTYPE ;
0DF8 00132 U.201: .WORD 3576 ;
          :TPASSUBEXP ;
0000* 00134 U.202: .WORD <<U.93-U.202>-2> ;
          :COLLOP1 ;
          00136 U.199: .BLKB 0 ;
          :TPASTYPE ;
102C 00136 U.203: .WORD 4140 ;
          :TPASTARGET ;
0000* 00138 U.205: .WORD <<U.204-U.205>-2> ;

```

```

0DF8 0013A ;TPASTYPE
          U.206: .WORD 3576 ;
0000* 0013C ;TPASSUBEXP
          U.207: .WORD <<U.93-U.207>-2> ;
          0013E ;COLLOP2
          U.204: .BLKB 0 ;
19F8 0013E ;TPASTYPE
          U.208: .WORD 6648 ;
0000* 00140 ;TPASSUBEXP
          U.210: .WORD <<U.209-U.210>-2> ;
0000* 00142 ;TPASTARGET
          U.212: .WORD <<U.211-U.212>-2> ;
0DF8 00144 ;TPASTYPE
          U.213: .WORD 3576 ;
0000* 00146 ;TPASSUBEXP
          U.214: .WORD <<U.93-U.214>-2> ;
          00148 ;COLLOP3
          U.211: .BLKB 0 ;
1DF8 00148 ;TPASTYPE
          U.215: .WORD 7672 ;
0000* 0014A ;TPASSUBEXP
          U.216: .WORD <<U.96-U.216>-2> ;
FFFF 0014C ;TPASTARGET
          U.217: .WORD -1 ;
          0014E ;COLLECT_PSECT_LIST
          U.209: .BLKB 0 ;
99F8 0014E ;TPASTYPE
          U.218: .WORD -26120 ;
0000* 00150 ;TPASSUBEXP
          U.219: .WORD <<U.124-U.219>-2> ;
00000000V 00152 ;TPASACTION
          U.220: .LONG <<COLLECT_PSECT-U.220>-4> ;
0000* 00156 ;TPASTARGET
          U.222: .WORD <<U.221-U.222>-2> ;
0DF8 00158 ;TPASTYPE
          U.223: .WORD 3576 ;
0000* 0015A ;TPASSUBEXP
          U.224: .WORD <<U.93-U.224>-2> ;
          0015C ;COLPSC1
          U.221: .BLKB 0 ;
102C 0015C ;TPASTYPE
          U.225: .WORD 4140 ;
0000* 0015E ;TPASTARGET
          U.226: .WORD <<U.209-U.226>-2> ;
11F7 00160 ;TPASTYPE
          U.227: .WORD 4599 ;
FFFF 00162 ;TPASTARGET
          U.228: .WORD -1 ;
0DF8 00164 ;TPASTYPE
          U.229: .WORD 3576 ;
0000* 00166 ;TPASSUBEXP
          U.230: .WORD <<U.93-U.230>-2> ;
          00168 ;DZRO_MIN_OPTION
          U.23: .BLKB 0 ;
0504 00168 ;TPASTYPE
          U.234: .WORD 1284 ;
143D 0016A ;TPASTYPE

```



```

0000* 0016C U.236: .WORD 5181 ;
          :TPASTARGET ;
          U.238: .WORD <<U.237-U.238>-2> ;
0016E :DZROMIN1 ;
          U.237: .BLKB 0 ;
99F8 0016E :TPASTYPE ;
          U.239: .WORD -26120 ;
0000* 00170 :TPASSUBEXP ;
          U.240: .WORD <<U.87-U.240>-2> ;
00000000V 00172 :TPASACTION ;
          U.241: .LONG <<SET_MIN_DZRO-U.241>-4> ;
0000* 00176 :TPASTARGET ;
          U.243: .WORD <<U.242-U.243>-2> ;
0DF8 00178 :TPASTYPE ;
          U.244: .WORD 3576 ;
0000* 0017A :TPASSUBEXP ;
          U.245: .WORD <<U.93-U.245>-2> ;
          0017C :DZROMIN2 ;
          U.242: .BLKB 0 ;
1DF8 0017C :TPASTYPE ;
          U.246: .WORD 7672 ;
0000* 0017E :TPASSUBEXP ;
          U.247: .WORD <<U.96-U.247>-2> ;
FFFF 00180 :TPASTARGET ;
          U.248: .WORD -1 ;
          00182 :GSMATCH_OPTION ;
          U.27: .BLKB 0 ;
0505 00182 :TPASTYPE ;
          U.252: .WORD 1285 ;
043D 00184 :TPASTYPE ;
          U.254: .WORD 1085 ;
19F8 00186 :TPASTYPE ;
          U.255: .WORD 6648 ;
0000* 00188 :TPASSUBEXP ;
          U.257: .WORD <<U.256-U.257>-2> ;
0000* 0018A :TPASTARGET ;
          U.259: .WORD <<U.258-U.259>-2> ;
0DF8 0018C :TPASTYPE ;
          U.260: .WORD 3576 ;
0000* 0018E :TPASSUBEXP ;
          U.261: .WORD <<U.93-U.261>-2> ;
          00190 :GSMOP1 ;
          U.258: .BLKB 0 ;
102C 00190 :TPASTYPE ;
          U.262: .WORD 4140 ;
0000* 00192 :TPASTARGET ;
          U.264: .WORD <<U.263-U.264>-2> ;
0DF8 00194 :TPASTYPE ;
          U.265: .WORD 3576 ;
0000* 00196 :TPASSUBEXP ;
          U.266: .WORD <<U.93-U.266>-2> ;
          00198 :GSMOP2 ;
          U.263: .BLKB 0 ;
99F8 00198 :TPASTYPE ;
          U.267: .WORD -26120 ;
0000* 0019A :TPASSUBEXP ;
          U.268: .WORD <<U.87-U.268>-2> ;

```

```

00000000V 0019C :TPASACTION
                U.269: .LONG    <<SET_GSMATCH_MAJ-U.269>-4>
0000* 001A0 :TPASTARGET
                U.271: .WORD    <<U.270-U.271>-2>
0DF8 001A2 :TPASTYPE
                U.272: .WORD    3576
0000* 001A4 :TPASSUBEXP
                U.273: .WORD    <<U.93-U.273>-2>
                001A6 :GSMOP3
                U.270: .BLKB    0
102C 001A6 :TPASTYPE
                U.274: .WORD    4140
0000* 001A8 :TPASTARGET
                U.276: .WORD    <<U.275-U.276>-2>
0DF8 001AA :TPASTYPE
                U.277: .WORD    3576
0000* 001AC :TPASSUBEXP
                U.278: .WORD    <<U.93-U.278>-2>
                001AE :GSMOP4
                U.275: .BLKB    0
99F8 001AE :TPASTYPE
                U.279: .WORD    -26120
0000* 001B0 :TPASSUBEXP
                U.280: .WORD    <<U.87-U.280>-2>
00000000V 001B2 :TPASACTION
                U.281: .LONG    <<SET_GSMATCH_MIN-U.281>-4>
0000* 001B6 :TPASTARGET
                U.283: .WORD    <<U.282-U.283>-2>
0DF8 001B8 :TPASTYPE
                U.284: .WORD    3576
0000* 001BA :TPASSUBEXP
                U.285: .WORD    <<U.93-U.285>-2>
                001BC :GSMOP5
                U.282: .BLKB    0
1DF8 001BC :TPASTYPE
                U.286: .WORD    7672
0000* 001BE :TPASSUBEXP
                U.287: .WORD    <<U.96-U.287>-2>
FFFF 001C0 :TPASTARGET
                U.288: .WORD    -1
                001C2 :GSMATCH_PARSE_CONTROL
                U.256: .BLKB    0
9306 001C2 :TPASTYPE
                U.292: .WORD    -27898
01 001C4 :TPASFLAGS2
                U.293: .BYTE    1
00000000 001C5 :TPASPARAM
                U.294: .LONG    0
00000000V 001C9 :TPASACTION
                U.295: .LONG    <<SET_GSMATCH_CTL-U.295>-4>
FFFF 001CD :TPASTARGET
                U.296: .WORD    -1
9307 001CF :TPASTYPE
                U.300: .WORD    -27897
01 001D1 :TPASFLAGS2
                U.301: .BYTE    1
00000001 001D2 :TPASPARAM

```



```

00000000V 001D6 ; U.302: .LONG 1 ;
; TPA$ACTION ;
FFFF 001DA ; U.303: .LONG <<SET_GSMATCH_CTL-U.303>-4> ;
; TPA$TARGET ;
9308 001DC ; U.304: .WORD -1 ;
; TPA$TYPE ;
01 001DE ; U.308: .WORD -27896 ;
; TPA$FLAGS2 ;
00000002 001DF ; U.309: .BYTE 1 ;
; TPA$PARAM ;
00000000V 001E3 ; U.310: .LONG 2 ;
; TPA$ACTION ;
FFFF 001E7 ; U.311: .LONG <<SET_GSMATCH_CTL-U.311>-4> ;
; TPA$TARGET ;
0DF8 001E9 ; U.312: .WORD -1 ;
; TPA$TYPE ;
0000* 001EB ; U.313: .WORD 3576 ;
; TPA$SUBEXP ;
001ED ; U.314: .WORD <<U.93-U.314>-2> ;
; IMAGE_TYPE OPTION ;
0509 001ED ; U.67: .BLKB 0 ;
; TPA$TYPE ;
043D 001EF ; U.319: .WORD 1289 ;
; TPA$TYPE ;
19F8 001F1 ; U.321: .WORD 1085 ;
; TPA$TYPE ;
0000* 001F3 ; U.322: .WORD 6648 ;
; TPA$SUBEXP ;
0000* 001F5 ; U.324: .WORD <<U.323-U.324>-2> ;
; TPA$TARGET ;
0DF8 001F7 ; U.326: .WORD <<U.325-U.326>-2> ;
; TPA$TYPE ;
0000* 001F9 ; U.327: .WORD 3576 ;
; TPA$SUBEXP ;
001FB ; U.328: .WORD <<U.93-U.328>-2> ;
; IMAGEOP1 ;
1DF8 001FB ; U.325: .BLKB 0 ;
; TPA$TYPE ;
0000* 001FD ; U.329: .WORD 7672 ;
; TPA$SUBEXP ;
FFFF 001FF ; U.330: .WORD <<U.96-U.330>-2> ;
; TPA$TARGET ;
00201 ; U.331: .WORD -1 ;
; IMAGE_TYPE PARSE ;
910A 00201 ; U.323: .BLKB 0 ;
; TPA$TYPE ;
00000000V 00203 ; U.335: .WORD -28406 ;
; TPA$ACTION ;
FFFF 00207 ; U.336: .LONG <<SET_IMAGE_TYPE-U.336>-4> ;
; TPA$TARGET ;
0DF8 00209 ; U.337: .WORD -1 ;
; TPA$TYPE ;
0000* 0020B ; U.338: .WORD 3576 ;
; TPA$SUBEXP ;
0020D ; U.339: .WORD <<U.93-U.339>-2> ;
; IMAGE_IDENT OPTION ;
; U.71: .BLKB 0 ;

```

050B	0020D	;TPASTYPE				
		U.344: .WORD	1291			:
043D	0020F	;TPASTYPE				:
		U.346: .WORD	1085			:
9222	00211	;TPASTYPE				:
		U.347: .WORD	-28126			:
01	00213	;TPASFLAGS2				:
		U.348: .BYTE	1			:
00000001	00214	;TPASPARAM				:
		U.349: .LONG	1			:
00000000V	00218	;TPASACTION				:
		U.350: .LONG	<<SET_IMAGE_IDENT-U.350>-4>			:
FFFF	0021C	;TPASTARGET				:
		U.351: .WORD	-1			:
93F1	0021E	;TPASTYPE				:
		U.352: .WORD	-27663			:
01	00220	;TPASFLAGS2				:
		U.353: .BYTE	1			:
FFFFFFFFE	00221	;TPASPARAM				:
		U.354: .LONG	-2			:
00000000V	00225	;TPASACTION				:
		U.355: .LONG	<<SET_IMAGE_IDENT-U.355>-4>			:
FFFF	00229	;TPASTARGET				:
		U.356: .WORD	-1			:
0DF8	0022B	;TPASTYPE				:
		U.357: .WORD	3576			:
0000*	0022D	;TPASSUBEXP				:
		U.358: .WORD	<<U.93-U.358>-2>			:
	0022F	;IMAGE_NAME OPTION				:
		U.75: .BLKB	0			:
050C	0022F	;TPASTYPE				:
		U.362: .WORD	1292			:
043D	00231	;TPASTYPE				:
		U.364: .WORD	1085			:
99F8	00233	;TPASTYPE				:
		U.365: .WORD	-26120			:
0000*	00235	;TPASSUBEXP				:
		U.366: .WORD	<<U.124-U.366>-2>			:
00000000V	00237	;TPASACTION				:
		U.367: .LONG	<<SET_IMAGE_NAME-U.367>-4>			:
0000*	0023B	;TPASTARGET				:
		U.369: .WORD	<<U.368-U.369>-2>			:
0DF8	0023D	;TPASTYPE				:
		U.370: .WORD	3576			:
0000*	0023F	;TPASSUBEXP				:
		U.371: .WORD	<<U.93-U.371>-2>			:
	00241	;NAMEOP1				:
		U.368: .BLKB	0			:
1DF8	00241	;TPASTYPE				:
		U.372: .WORD	7672			:
0000*	00243	;TPASSUBEXP				:
		U.373: .WORD	<<U.96-U.373>-2>			:
FFFF	00245	;TPASTARGET				:
		U.374: .WORD	-1			:
	00247	;IOSEGMENT OPTION				:
		U.31: .BLKB	0			:
050D	00247	;TPASTYPE				:

043D	00249	U.378: .WORD	1293	:
		:TPASTYPE		:
99F8	0024B	U.380: .WORD	1085	:
		:TPASTYPE		:
0000*	0024D	U.381: .WORD	-26120	:
		:TPASSUBEXP		:
00000000V	0024F	U.382: .WORD	<<U.87-U.382>-2>	:
		:TPASACTION		:
0000*	00253	U.383: .LONG	<<SET_IOSEG-U.383>-4>	:
		:TPASTARGET		:
0DF8	00255	U.385: .WORD	<<U.384-U.385>-2>	:
		:TPASTYPE		:
0000*	00257	U.386: .WORD	3576	:
		:TPASSUBEXP		:
	00259	U.387: .WORD	<<U.93-U.387>-2>	:
		:IOSEGOP1		:
11F7	00259	U.384: .BLKB	0	:
		:TPASTYPE		:
FFFF	0025B	U.388: .WORD	4599	:
		:TPASTARGET		:
102C	0025D	U.389: .WORD	-1	:
		:TPASTYPE		:
0000*	0025F	U.390: .WORD	4140	:
		:TPASTARGET		:
0DF8	00261	U.392: .WORD	<<U.391-U.392>-2>	:
		:TPASTYPE		:
0000*	00263	U.393: .WORD	3576	:
		:TPASSUBEXP		:
	00265	U.394: .WORD	<<U.93-U.394>-2>	:
		:IOSEGOP2		:
930E	00265	U.391: .BLKB	0	:
		:TPASTYPE		:
01	00267	U.398: .WORD	-27890	:
		:TPASFLAGS2		:
00000000	00268	U.399: .BYTE	1	:
		:TPASPARAM		:
00000000V	0026C	U.400: .LONG	0	:
		:TPASACTION		:
FFFF	00270	U.401: .LONG	<<SET_POBUFS-U.401>-4>	:
		:TPASTARGET		:
930F	00272	U.402: .WORD	-1	:
		:TPASTYPE		:
01	00274	U.406: .WORD	-27889	:
		:TPASFLAGS2		:
00000001	00275	U.407: .BYTE	1	:
		:TPASPARAM		:
00000000V	00279	U.408: .LONG	1	:
		:TPASACTION		:
FFFF	0027D	U.409: .LONG	<<SET_POBUFS-U.409>-4>	:
		:TPASTARGET		:
0DF8	0027F	U.410: .WORD	-1	:
		:TPASTYPE		:
0000*	00281	U.411: .WORD	3576	:
		:TPASSUBEXP		:
	00283	U.412: .WORD	<<U.93-U.412>-2>	:
		:ISD_MAX_OPTION		:
		U.35: .BLKB	0	:

0510	00283	:TPASTYPE			
		U.417: .WORD	1296		:
043D	00285	:TPASTYPE			
		U.419: .WORD	1085		:
99F8	00287	:TPASTYPE			
		U.420: .WORD	-26120		:
0000*	00289	:TPASSUBEXP			
		U.421: .WORD	<<U.87-U.421>-2>		:
00000000V	0028B	:TPASACTION			
		U.422: .LONG	<<SET_ISD_MAX-U.422>-4>		:
0000*	0028F	:TPASTARGET			
		U.424: .WORD	<<U.423-U.424>-2>		:
0DF8	00291	:TPASTYPE			
		U.425: .WORD	3576		:
0000*	00293	:TPASSUBEXP			
		U.426: .WORD	<<U.93-U.426>-2>		:
	00295	:ISD_MAXOP1			
		U.423: .BLKB	0		:
1DF8	00295	:TPASTYPE			
		U.427: .WORD	7672		:
0000*	00297	:TPASSUBEXP			
		U.428: .WORD	<<U.96-U.428>-2>		:
FFFF	00299	:TPASTARGET			
		U.429: .WORD	-1		:
	0029B	:PROTECT_OPTION			
		U.39: .BLKB	0		:
0511	0029B	:TPASTYPE			
		U.433: .WORD	1297		:
043D	0029D	:TPASTYPE			
		U.435: .WORD	1085		:
9312	0029F	:TPASTYPE			
		U.439: .WORD	-27886		:
01	002A1	:TPASFLAGS2			
		U.440: .BYTE	1		:
00000001	002A2	:TPASPARAM			
		U.441: .LONG	1		:
00000000V	002A6	:TPASACTION			
		U.442: .LONG	<<SET_PROTECT-U.442>-4>		:
0000*	002AA	:TPASTARGET			
		U.444: .WORD	<<U.443-U.444>-2>		:
9313	002AC	:TPASTYPE			
		U.448: .WORD	-27885		:
01	002AE	:TPASFLAGS2			
		U.449: .BYTE	1		:
00000000	002AF	:TPASPARAM			
		U.450: .LONG	0		:
00000000V	002B3	:TPASACTION			
		U.451: .LONG	<<SET_PROTECT-U.451>-4>		:
0000*	002B7	:TPASTARGET			
		U.452: .WORD	<<U.443-U.452>-2>		:
0DF8	002B9	:TPASTYPE			
		U.453: .WORD	3576		:
0000*	002BB	:TPASSUBEXP			
		U.454: .WORD	<<U.93-U.454>-2>		:
	002BD	:PROTOP1			
		U.443: .BLKB	0		:
1DF8	002BD	:TPASTYPE			


```

0000* 002BF U.456: .WORD 7672
          ;TPASSUBEXP
FFFF 002C1 U.457: .WORD <<U.96-U.457>-2>
          ;TPASTARGET
          U.458: .WORD -1
          002C3 ;PSECT_OPTION
          U.43: .BLKB 0
0514 002C3 ;TPASTYPE
          U.462: .WORD 1300
043D 002C5 ;TPASTYPE
          U.464: .WORD 1085
99F8 002C7 ;TPASTYPE
          U.465: .WORD -26120
0000* 002C9 ;TPASSUBEXP
          U.466: .WORD <<U.124-U.466>-2>
00000000V 002CB ;TPSACTION
          U.467: .LONG <<CREATEPSECT-U.467>-4>
0000* 002CF ;TPASTARGET
          U.469: .WORD <<U.468-U.469>-2>
0DF8 002D1 ;TPASTYPE
          U.470: .WORD 3576
0000* 002D3 ;TPASSUBEXP
          U.471: .WORD <<U.93-U.471>-2>
          002D5 ;PSCOP1
          U.468: .BLKB 0
102C 002D5 ;TPASTYPE
          U.472: .WORD 4140
0000* 002D7 ;TPASTARGET
          U.474: .WORD <<U.473-U.474>-2>
0DF8 002D9 ;TPASTYPE
          U.475: .WORD 3576
0000* 002DB ;TPASSUBEXP
          U.476: .WORD <<U.93-U.476>-2>
          002DD ;PSCOP2
          U.473: .BLKB 0
1DF8 002DD ;TPASTYPE
          U.477: .WORD 7672
0000* 002DF ;TPASSUBEXP
          U.479: .WORD <<U.478-U.479>-2>
0000* 002E1 ;TPASTARGET
          U.481: .WORD <<U.480-U.481>-2>
          002E3 ;PSCOP3
          U.480: .BLKB 0
102C 002E3 ;TPASTYPE
          U.482: .WORD 4140
0000* 002E5 ;TPASTARGET
          U.483: .WORD <<U.473-U.483>-2>
11F7 002E7 ;TPASTYPE
          U.484: .WORD 4599
FFFF 002E9 ;TPASTARGET
          U.485: .WORD -1
0DF8 002EB ;TPASTYPE
          U.486: .WORD 3576
0000* 002ED ;TPASSUBEXP
          U.487: .WORD <<U.93-U.487>-2>
          002EF ;PSECT_PARSE_ATTRIBUTE
          U.478: .BLKB 0

```

9315	002EF	;TPASTYPE				
		U.491: .WORD	-27883			:
01	002F1	;TPAS\$FLAGS2				:
		U.492: .BYTE	1			:
0000FFF7	002F2	;TPAS\$PARAM				:
		U.493: .LONG	65527			:
00000000V	002F6	;TPAS\$ACTION				:
		U.494: .LONG	<<SET_PSCATRIB-U.494>-4>			:
FFFF	002FA	;TPAS\$TARGET				:
		U.495: .WORD	-1			:
9316	002FC	;TPASTYPE				:
		U.499: .WORD	-27882			:
01	002FE	;TPAS\$FLAGS2				:
		U.500: .BYTE	1			:
00000008	002FF	;TPAS\$PARAM				:
		U.501: .LONG	8			:
00000000V	00303	;TPAS\$ACTION				:
		U.502: .LONG	<<SET_PSCATRIB-U.502>-4>			:
FFFF	00307	;TPAS\$TARGET				:
		U.503: .WORD	-1			:
9317	00309	;TPASTYPE				:
		U.507: .WORD	-27881			:
01	0030B	;TPAS\$FLAGS2				:
		U.508: .BYTE	1			:
0000FFFB	0030C	;TPAS\$PARAM				:
		U.509: .LONG	65531			:
00000000V	00310	;TPAS\$ACTION				:
		U.510: .LONG	<<SET_PSCATRIB-U.510>-4>			:
FFFF	00314	;TPAS\$TARGET				:
		U.511: .WORD	-1			:
9318	00316	;TPASTYPE				:
		U.515: .WORD	-27880			:
01	00318	;TPAS\$FLAGS2				:
		U.516: .BYTE	1			:
00000004	00319	;TPAS\$PARAM				:
		U.517: .LONG	4			:
00000000V	0031D	;TPAS\$ACTION				:
		U.518: .LONG	<<SET_PSCATRIB-U.518>-4>			:
FFFF	00321	;TPAS\$TARGET				:
		U.519: .WORD	-1			:
9319	00323	;TPASTYPE				:
		U.523: .WORD	-27879			:
01	00325	;TPAS\$FLAGS2				:
		U.524: .BYTE	1			:
000000C0	00326	;TPAS\$PARAM				:
		U.525: .LONG	192			:
00000000V	0032A	;TPAS\$ACTION				:
		U.526: .LONG	<<SET_PSCATRIB-U.526>-4>			:
FFFF	0032E	;TPAS\$TARGET				:
		U.527: .WORD	-1			:
931A	00330	;TPASTYPE				:
		U.531: .WORD	-27878			:
01	00332	;TPAS\$FLAGS2				:
		U.532: .BYTE	1			:
0000FFBF	00333	;TPAS\$PARAM				:
		U.533: .LONG	65471			:
00000000V	00337	;TPAS\$ACTION				:

		U.534: .LONG	<<SET_PSCATRIB-U.534>-4>	:
FFFF	0033B	:TPASTARGET		:
		U.535: .WORD	-1	:
931B	0033D	:TPASTYPE		:
		U.539: .WORD	-27877	:
01	0033F	:TPASFLAGS2		:
		U.540: .BYTE	1	:
00000010	00340	:TPASPARAM		:
		U.541: .LONG	16	:
00000000V	00344	:TPASACTION		:
		U.542: .LONG	<<SET_PSCATRIB-U.542>-4>	:
FFFF	00348	:TPASTARGET		:
		U.543: .WORD	-1	:
931C	0034A	:TPASTYPE		:
		U.547: .WORD	-27876	:
01	0034C	:TPASFLAGS2		:
		U.548: .BYTE	1	:
0000FFEF	0034D	:TPASPARAM		:
		U.549: .LONG	65519	:
00000000V	00351	:TPASACTION		:
		U.550: .LONG	<<SET_PSCATRIB-U.550>-4>	:
FFFF	00355	:TPASTARGET		:
		U.551: .WORD	-1	:
931D	00357	:TPASTYPE		:
		U.555: .WORD	-27875	:
01	00359	:TPASFLAGS2		:
		U.556: .BYTE	1	:
00000002	0035A	:TPASPARAM		:
		U.557: .LONG	2	:
00000000V	0035E	:TPASACTION		:
		U.558: .LONG	<<SET_PSCATRIB-U.558>-4>	:
FFFF	00362	:TPASTARGET		:
		U.559: .WORD	-1	:
931E	00364	:TPASTYPE		:
		U.563: .WORD	-27874	:
01	00366	:TPASFLAGS2		:
		U.564: .BYTE	1	:
0000FFFD	00367	:TPASPARAM		:
		U.565: .LONG	65533	:
00000000V	0036B	:TPASACTION		:
		U.566: .LONG	<<SET_PSCATRIB-U.566>-4>	:
FFFF	0036F	:TPASTARGET		:
		U.567: .WORD	-1	:
931F	00371	:TPASTYPE		:
		U.571: .WORD	-27873	:
01	00373	:TPASFLAGS2		:
		U.572: .BYTE	1	:
00000001	00374	:TPASPARAM		:
		U.573: .LONG	1	:
00000000V	00378	:TPASACTION		:
		U.574: .LONG	<<SET_PSCATRIB-U.574>-4>	:
FFFF	0037C	:TPASTARGET		:
		U.575: .WORD	-1	:
9320	0037E	:TPASTYPE		:
		U.579: .WORD	-27872	:
01	00380	:TPASFLAGS2		:
		U.580: .BYTE	1	:

0000FFFE	00381	:TPASPARAM			
		U.581: .LONG	65534		:
00000000V	00385	:TPASACTION			:
		U.582: .LONG	<<SET_PSCATRIB-U.582>-4>		:
FFFF	00389	:TPASTARGET			:
		U.583: .WORD	-1		:
9321	0038B	:TPASTYPE			:
		U.587: .WORD	-27871		:
01	0038D	:TPASFLAGS2			:
		U.588: .BYTE	1		:
00000080	0038E	:TPASPARAM			:
		U.589: .LONG	128		:
00000000V	00392	:TPASACTION			:
		U.590: .LONG	<<SET_PSCATRIB-U.590>-4>		:
FFFF	00396	:TPASTARGET			:
		U.591: .WORD	-1		:
9322	00398	:TPASTYPE			:
		U.595: .WORD	-27870		:
01	0039A	:TPASFLAGS2			:
		U.596: .BYTE	1		:
0000FF7F	0039B	:TPASPARAM			:
		U.597: .LONG	65407		:
00000000V	0039F	:TPASACTION			:
		U.598: .LONG	<<SET_PSCATRIB-U.598>-4>		:
FFFF	003A3	:TPASTARGET			:
		U.599: .WORD	-1		:
9323	003A5	:TPASTYPE			:
		U.603: .WORD	-27869		:
01	003A7	:TPASFLAGS2			:
		U.604: .BYTE	1		:
00000020	003A8	:TPASPARAM			:
		U.605: .LONG	32		:
00000000V	003AC	:TPASACTION			:
		U.606: .LONG	<<SET_PSCATRIB-U.606>-4>		:
FFFF	003B0	:TPASTARGET			:
		U.607: .WORD	-1		:
9324	003B2	:TPASTYPE			:
		U.611: .WORD	-27868		:
01	003B4	:TPASFLAGS2			:
		U.612: .BYTE	1		:
0000FFDF	003B5	:TPASPARAM			:
		U.613: .LONG	65503		:
00000000V	003B9	:TPASACTION			:
		U.614: .LONG	<<SET_PSCATRIB-U.614>-4>		:
FFFF	003BD	:TPASTARGET			:
		U.615: .WORD	-1		:
9325	003BF	:TPASTYPE			:
		U.619: .WORD	-27867		:
01	003C1	:TPASFLAGS2			:
		U.620: .BYTE	1		:
00000180	003C2	:TPASPARAM			:
		U.621: .LONG	384		:
00000000V	003C6	:TPASACTION			:
		U.622: .LONG	<<SET_PSCATRIB-U.622>-4>		:
FFFF	003CA	:TPASTARGET			:
		U.623: .WORD	-1		:
9326	003CC	:TPASTYPE			:

01	003CE	U.627: .WORD	-27866	:
		:TPAS\$FLAGS2		:
0000FEFF	003CF	U.628: .BYTE	1	:
		:TPAS\$PARAM		:
00000000V	003D3	U.629: .LONG	65279	:
		:TPAS\$ACTION		:
FFFF	003D7	U.630: .LONG	<<SET_PSCATTRIB-U.630>-4>	:
		:TPAS\$TARGET		:
9327	003D9	U.631: .WORD	-1	:
		:TPAS\$TYPE		:
01	003DB	U.635: .WORD	-27865	:
		:TPAS\$FLAGS2		:
00000200	003DC	U.636: .BYTE	1	:
		:TPAS\$PARAM		:
00000000V	003E0	U.637: .LONG	512	:
		:TPAS\$ACTION		:
FFFF	003E4	U.638: .LONG	<<SET_PSCATTRIB-U.638>-4>	:
		:TPAS\$TARGET		:
9328	003E6	U.639: .WORD	-1	:
		:TPAS\$TYPE		:
01	003E8	U.643: .WORD	-27864	:
		:TPAS\$FLAGS2		:
0000FDFF	003E9	U.644: .BYTE	1	:
		:TPAS\$PARAM		:
00000000V	003ED	U.645: .LONG	65023	:
		:TPAS\$ACTION		:
FFFF	003F1	U.646: .LONG	<<SET_PSCATTRIB-U.646>-4>	:
		:TPAS\$TARGET		:
9329	003F3	U.647: .WORD	-1	:
		:TPAS\$TYPE		:
01	003F5	U.651: .WORD	-27863	:
		:TPAS\$FLAGS2		:
80000000	003F6	U.652: .BYTE	1	:
		:TPAS\$PARAM		:
00000000V	003FA	U.653: .LONG	-2147483648	:
		:TPAS\$ACTION		:
FFFF	003FE	U.654: .LONG	<<SET_PSCATTRIB-U.654>-4>	:
		:TPAS\$TARGET		:
932A	00400	U.655: .WORD	-1	:
		:TPAS\$TYPE		:
01	00402	U.659: .WORD	-27862	:
		:TPAS\$FLAGS2		:
80000001	00403	U.660: .BYTE	1	:
		:TPAS\$PARAM		:
00000000V	00407	U.661: .LONG	-2147483647	:
		:TPAS\$ACTION		:
FFFF	0040B	U.662: .LONG	<<SET_PSCATTRIB-U.662>-4>	:
		:TPAS\$TARGET		:
932B	0040D	U.663: .WORD	-1	:
		:TPAS\$TYPE		:
01	0040F	U.667: .WORD	-27861	:
		:TPAS\$FLAGS2		:
80000002	00410	U.668: .BYTE	1	:
		:TPAS\$PARAM		:
00000000V	00414	U.669: .LONG	-2147483646	:
		:TPAS\$ACTION		:
		U.670: .LONG	<<SET_PSCATTRIB-U.670>-4>	:

FFFF	00418	:TPASTARGET				
		U.671: .WORD	-1			:
932C	0041A	:TPASTYPE				:
		U.675: .WORD	-27860			:
01	0041C	:TPASFLAGS2				:
		U.676: .BYTE	1			:
80000003	0041D	:TPASPARAM				:
		U.677: .LONG	-2147483645			:
00000000V	00421	:TPASACTION				:
		U.678: .LONG	<<SET_PSCATRIB-U.678>-4>			:
FFFF	00425	:TPASTARGET				:
		U.679: .WORD	-1			:
932D	00427	:TPASTYPE				:
		U.683: .WORD	-27859			:
01	00429	:TPASFLAGS2				:
		U.684: .BYTE	1			:
80000009	0042A	:TPASPARAM				:
		U.685: .LONG	-2147483639			:
00000000V	0042E	:TPASACTION				:
		U.686: .LONG	<<SET_PSCATRIB-U.686>-4>			:
FFFF	00432	:TPASTARGET				:
		U.687: .WORD	-1			:
9230	00434	:TPASTYPE				:
		U.688: .WORD	-28112			:
01	00436	:TPASFLAGS2				:
		U.689: .BYTE	1			:
80000000	00437	:TPASPARAM				:
		U.690: .LONG	-2147483648			:
00000000V	0043B	:TPASACTION				:
		U.691: .LONG	<<SET_PSCATRIB-U.691>-4>			:
FFFF	0043F	:TPASTARGET				:
		U.692: .WORD	-1			:
9231	00441	:TPASTYPE				:
		U.693: .WORD	-28111			:
01	00443	:TPASFLAGS2				:
		U.694: .BYTE	1			:
80000001	00444	:TPASPARAM				:
		U.695: .LONG	-2147483647			:
00000000V	00448	:TPASACTION				:
		U.696: .LONG	<<SET_PSCATRIB-U.696>-4>			:
FFFF	0044C	:TPASTARGET				:
		U.697: .WORD	-1			:
9232	0044E	:TPASTYPE				:
		U.698: .WORD	-28110			:
01	00450	:TPASFLAGS2				:
		U.699: .BYTE	1			:
80000002	00451	:TPASPARAM				:
		U.700: .LONG	-2147483646			:
00000000V	00455	:TPASACTION				:
		U.701: .LONG	<<SET_PSCATRIB-U.701>-4>			:
FFFF	00459	:TPASTARGET				:
		U.702: .WORD	-1			:
9233	0045B	:TPASTYPE				:
		U.703: .WORD	-28109			:
01	0045D	:TPASFLAGS2				:
		U.704: .BYTE	1			:
80000003	0045E	:TPASPARAM				:

00000000V	00462	U.705: .LONG	-2147483645	:
		:TPASACTION		:
FFFF	00466	U.706: .LONG	<<SET_PSCATRIB-U.706>-4>	:
		:TPASTARGET		:
9239	00468	U.707: .WORD	-1	:
		:TPASTYPE		:
01	0046A	U.708: .WORD	-28103	:
		:TPASFLAGS2		:
80000009	0046B	U.709: .BYTE	1	:
		:TPASPARAM		:
00000000V	0046F	U.710: .LONG	-2147483639	:
		:TPASACTION		:
FFFF	00473	U.711: .LONG	<<SET_PSCATRIB-U.711>-4>	:
		:TPASTARGET		:
0DF8	00475	U.712: .WORD	-1	:
		:TPASTYPE		:
0000*	00477	U.713: .WORD	3576	:
		:TPASSUBEXP		:
	00479	U.714: .WORD	<<U.93-U.714>-2>	:
		:SHL_OPTION		:
052E	00479	U.47: .BLKB	0	:
		:TPASTYPE		:
043D	0047B	U.719: .WORD	1326	:
		:TPASTYPE		:
99F8	0047D	U.721: .WORD	1085	:
		:TPASTYPE		:
0000*	0047F	U.722: .WORD	-26120	:
		:TPASSUBEXP		:
00000000V	00481	U.723: .WORD	<<U.87-U.723>-2>	:
		:TPASACTION		:
0000*	00485	U.724: .LONG	<<SET_SHL-U.724>-4>	:
		:TPASTARGET		:
0DF8	00487	U.726: .WORD	<<U.725-U.726>-2>	:
		:TPASTYPE		:
0000*	00489	U.727: .WORD	3576	:
		:TPASSUBEXP		:
	0048B	U.728: .WORD	<<U.93-U.728>-2>	:
		:SHL1		:
1DF8	0048B	U.725: .BLKB	0	:
		:TPASTYPE		:
0000*	0048D	U.729: .WORD	7672	:
		:TPASSUBEXP		:
FFFF	0048F	U.730: .WORD	<<U.96-U.730>-2>	:
		:TPASTARGET		:
	00491	U.731: .WORD	-1	:
		:STACK_OPTION		:
052F	00491	U.51: .BLKB	0	:
		:TPASTYPE		:
043D	00493	U.735: .WORD	1327	:
		:TPASTYPE		:
99F8	00495	U.737: .WORD	1085	:
		:TPASTYPE		:
0000*	00497	U.738: .WORD	-26120	:
		:TPASSUBEXP		:
00000000V	00499	U.739: .WORD	<<U.87-U.739>-2>	:
		:TPASACTION		:
		U.740: .LONG	<<SET_STACK-U.740>-4>	:


```

0000* 0049D :TPASTARGET
          U.742: .WORD <<U.741-U.742>-2> ;
0DF8 0049F :TPASTYPE
          U.743: .WORD 3576 ;
0000* 004A1 :TPASSUBEXP
          U.744: .WORD <<U.93-U.744>-2> ;
          004A3 :STACKOP1
          U.741: .BLKB 0 ;
1DF8 004A3 :TPASTYPE
          U.745: .WORD 7672 ;
0000* 004A5 :TPASSUBEXP
          U.746: .WORD <<U.96-U.746>-2> ;
FFFF 004A7 :TPASTARGET
          U.747: .WORD -1 ;
          004A9 :SYMBOL_OPTION
          U.55: .BLKB 0 ;
0530 004A9 :TPASTYPE
          U.751: .WORD 1328 ;
043D 004AB :TPASTYPE
          U.753: .WORD 1085 ;
99F8 004AD :TPASTYPE
          U.754: .WORD -26120 ;
0000* 004AF :TPASSUBEXP
          U.755: .WORD <<U.124-U.755>-2> ;
00000000V 004B1 :TPASACTION
          U.756: .LONG <<DEFINESYMBOLNAME-U.756>-4> ;
0000* 004B5 :TPASTARGET
          U.758: .WORD <<U.757-U.758>-2> ;
0DF8 004B7 :TPASTYPE
          U.759: .WORD 3576 ;
0000* 004B9 :TPASSUBEXP
          U.760: .WORD <<U.93-U.760>-2> ;
          004BB :SYMBOLOP1
          U.757: .BLKB 0 ;
102C 004BB :TPASTYPE
          U.761: .WORD 4140 ;
0000* 004BD :TPASTARGET
          U.763: .WORD <<U.762-U.763>-2> ;
0DF8 004BF :TPASTYPE
          U.764: .WORD 3576 ;
0000* 004C1 :TPASSUBEXP
          U.765: .WORD <<U.93-U.765>-2> ;
          004C3 :SYMBOLOP2
          U.762: .BLKB 0 ;
99F8 004C3 :TPASTYPE
          U.766: .WORD -26120 ;
0000* 004C5 :TPASSUBEXP
          U.767: .WORD <<U.87-U.767>-2> ;
00000000V 004C7 :TPASACTION
          U.768: .LONG <<DEFINESYMBOLVAL-U.768>-4> ;
0000* 004CB :TPASTARGET
          U.770: .WORD <<U.769-U.770>-2> ;
0DF8 004CD :TPASTYPE
          U.771: .WORD 3576 ;
0000* 004CF :TPASSUBEXP
          U.772: .WORD <<U.93-U.772>-2> ;
          004D1 :SYMBOLOP3

```


1DF8	004D1	U.769: .BLKB	0	
		:TPASTYPE		
0000*	004D3	U.773: .WORD	7672	:
		:TPASSUBEXP		
FFFF	004D5	U.774: .WORD	<<U.96-U.774>-2>	:
		:TPASTARGET		
	004D7	U.775: .WORD	-1	:
		:UNSUPPORTED OPTION		
0531	004D7	U.63: .BLKB	0	
		:TPASTYPE		
143D	004D9	U.779: .WORD	1329	:
		:TPASTYPE		
0000*	004DB	U.781: .WORD	5181	:
		:TPASTARGET		
	004DD	U.783: .WORD	<<U.782-U.783>-2>	:
		:UNSUPPOP1		
99F8	004DD	U.782: .BLKB	0	
		:TPASTYPE		
0000*	004DF	U.784: .WORD	-26120	:
		:TPASSUBEXP		
00000000V	004E1	U.785: .WORD	<<U.87-U.785>-2>	:
		:TPASACTION		
0000*	004E5	U.786: .LONG	<<SET_UNSUPPORTED-U.786>-4>	:
		:TPASTARGET		
0DF8	004E7	U.788: .WORD	<<U.787-U.788>-2>	:
		:TPASTYPE		
0000*	004E9	U.789: .WORD	3576	:
		:TPASSUBEXP		
	004EB	U.790: .WORD	<<U.93-U.790>-2>	:
		:UNSUPPOP2		
1DF8	004EB	U.787: .BLKB	0	
		:TPASTYPE		
0000*	004ED	U.791: .WORD	7672	:
		:TPASSUBEXP		
FFFF	004EF	U.792: .WORD	<<U.96-U.792>-2>	:
		:TPASTARGET		
	004F1	U.793: .WORD	-1	:
		:UNIVERSAL OPTION		
0532	004F1	U.59: .BLKB	0	
		:TPASTYPE		
043D	004F3	U.797: .WORD	1330	:
		:TPASTYPE		
1DF8	004F5	U.799: .WORD	1085	:
		:TPASTYPE		
0000*	004F7	U.800: .WORD	7672	:
		:TPASSUBEXP		
FFFF	004F9	U.802: .WORD	<<U.801-U.802>-2>	:
		:TPASTARGET		
	004FB	U.803: .WORD	-1	:
		:PARSE_UNIV SYMBOLS		
102A	004FB	U.801: .BLKB	0	
		:TPASTYPE		
0000*	004FD	U.804: .WORD	4138	:
		:TPASTARGET		
99F8	004FF	U.806: .WORD	<<U.805-U.806>-2>	:
		:TPASTYPE		
		U.807: .WORD	-26120	:

```

0000* 00501 :TPASSUBEXP
U.808: .WORD <<U.124-U.808>-2> ;
00000000V 00503 :TPASACTION
U.809: .LONG <<SET_UNIVERSAL-U.809>-4> ;
0000* 00507 :TPASTARGET
U.811: .WORD <<U.810-U.811>-2> ;
0DF8 00509 :TPASTYPE
U.812: .WORD 3576 ;
0000* 0050B :TPASSUBEXP
U.813: .WORD <<U.93-U.813>-2> ;
0050D :PRSUNIV1
U.810: .BLKB 0 ;
102C 0050D :TPASTYPE
U.814: .WORD 4140 ;
0000* 0050F :TPASTARGET
U.815: .WORD <<U.801-U.815>-2> ;
11F7 00511 :TPASTYPE
U.816: .WORD 4599 ;
FFFF 00513 :TPASTARGET
U.817: .WORD -1 ;
0DF8 00515 :TPASTYPE
U.818: .WORD 3576 ;
0000* 00517 :TPASSUBEXP
U.819: .WORD <<U.93-U.819>-2> ;
00519 :PRSUNIV2
U.805: .BLKB 0 ;
11F7 00519 :TPASTYPE
U.820: .WORD 4599 ;
0000* 0051B :TPASTARGET
U.822: .WORD <<U.821-U.822>-2> ;
0DF8 0051D :TPASTYPE
U.823: .WORD 3576 ;
0000* 0051F :TPASSUBEXP
U.824: .WORD <<U.93-U.824>-2> ;
00521 :PRSUNIV3
U.821: .BLKB 0 ;
95F6 00521 :TPASTYPE
U.825: .WORD -27146 ;
00000000V 00523 :TPASACTION
U.826: .LONG <<SET_ALLUNIV-U.826>-4> ;
FFFF 00527 :TPASTARGET
U.827: .WORD -1 ;
00529 :PARSE_FILELIST
U.3: .BLKB 0 ;
5DF8 00529 :TPASTYPE
U.828: .WORD 24056 ;
0000* 0052B :TPASSUBEXP
U.830: .WORD <<U.829-U.830>-2> ;
00000000* 0052D :TPASADDR
U.831: .LONG <<FILESPEC_DESC-U.831>-4> ;
0000* 00531 :TPASTARGET
U.833: .WORD <<U.832-U.833>-2> ;
00533 :PARSEFL1
U.832: .BLKB 0 ;
105F 00533 :TPASTYPE
U.834: .WORD 4191 ;
FFFE 00535 :TPASTARGET

```


103D	00537	U.835: .WORD	-2	:
		:TPASTYPE		
FFFE	00539	U.836: .WORD	4157	:
		:TPASTARGET		
902C	0053B	U.837: .WORD	-2	:
		:TPASTYPE		
00000000V	0053D	U.838: .WORD	-28628	:
		:TPASACTION		
0000*	00541	U.839: .LONG	<<PROCESSFILE-U.839>-4>	:
		:TPASTARGET		
102F	00543	U.840: .WORD	<<U.3-U.840>-2>	:
		:TPASTYPE		
0000*	00545	U.841: .WORD	4143	:
		:TPASTARGET		
91F7	00547	U.843: .WORD	<<U.842-U.843>-2>	:
		:TPASTYPE		
00000000V	00549	U.844: .WORD	-28169	:
		:TPASACTION		
FFFF	0054D	U.845: .LONG	<<PROCESSFILE-U.845>-4>	:
		:TPASTARGET		
0DF8	0054F	U.846: .WORD	-1	:
		:TPASTYPE		
0000*	00551	U.847: .WORD	3576	:
		:TPASSUBEXP		
		U.848: .WORD	<<U.93-U.848>-2>	:
	00553	:PARSEFL2		
9BF8	00553	U.842: .BLKB	0	:
		:TPASTYPE		
01	00555	U.849: .WORD	-25608	:
		:TPASFLAGS2		
0000*	00556	U.850: .BYTE	1	:
		:TPASSUBEXP		
		U.852: .WORD	<<U.851-U.852>-2>	:
0000001C	00558	:TPASPARAM		
		U.853: .LONG	28	:
00000000V	0055C	:TPASACTION		
		U.854: .LONG	<<SET_INCLUDE-U.854>-4>	:
0000*	00560	:TPASTARGET		
		U.856: .WORD	<<U.855-U.856>-2>	:
9333	00562	:TPASTYPE		
		U.860: .WORD	-27853	:
01	00564	:TPASFLAGS2		
		U.861: .BYTE	1	:
0000001C	00565	:TPASPARAM		
		U.862: .LONG	28	:
00000000V	00569	:TPASACTION		
		U.863: .LONG	<<SET_LIBRARY-U.863>-4>	:
0000*	0056D	:TPASTARGET		
		U.864: .WORD	<<U.855-U.864>-2>	:
9334	0056F	:TPASTYPE		
		U.868: .WORD	-27852	:
01	00571	:TPASFLAGS2		
		U.869: .BYTE	1	:
00000012	00572	:TPASPARAM		
		U.870: .LONG	18	:
00000000V	00576	:TPASACTION		
		U.871: .LONG	<<SET_SELECTIVE-U.871>-4>	:

0000*	0057A	:TPASTARGET			
		U.872: .WORD	<<U.855-U.872>-2>		:
9BF8	0057C	:TPASTYPE			
		U.873: .WORD	-25608		:
01	0057E	:TPASFLAGS2			
		U.874: .BYTE	1		:
0000*	0057F	:TPASSUBEXP			
		U.876: .WORD	<<U.875-U.876>-2>		:
00000002	00581	:TPASPARAM			
		U.877: .LONG	2		:
00000000V	00585	:TPASACTION			
		U.878: .LONG	<<SET_SHAREABLE-U.878>-4>		:
0000*	00589	:TPASTARGET			
		U.879: .WORD	<<U.855-U.879>-2>		:
0DF8	0058B	:TPASTYPE			
		U.880: .WORD	3576		:
0000*	0058D	:TPASSUBEXP			
		U.881: .WORD	<<U.93-U.881>-2>		:
	0058F	:PARSEFL3			
		U.855: .BLKB	0		:
102F	0058F	:TPASTYPE			
		U.883: .WORD	4143		:
0000*	00591	:TPASTARGET			
		U.884: .WORD	<<U.842-U.884>-2>		:
15F6	00593	:TPASTYPE			
		U.885: .WORD	5622		:
0000*	00595	:TPASTARGET			
		U.887: .WORD	<<U.886-U.887>-2>		:
	00597	:PARSEFL4			
		U.886: .BLKB	0		:
902C	00597	:TPASTYPE			
		U.888: .WORD	-28628		:
00000000V	00599	:TPASACTION			
		U.889: .LONG	<<PROCESSFILE-U.889>-4>		:
0000*	0059D	:TPASTARGET			
		U.890: .WORD	<<U.3-U.890>-2>		:
91F7	0059F	:TPASTYPE			
		U.891: .WORD	-28169		:
00000000V	005A1	:TPASACTION			
		U.892: .LONG	<<PROCESSFILE-U.892>-4>		:
FFFF	005A5	:TPASTARGET			
		U.893: .WORD	-1		:
0DF8	005A7	:TPASTYPE			
		U.894: .WORD	3576		:
0000*	005A9	:TPASSUBEXP			
		U.895: .WORD	<<U.93-U.895>-2>		:
	005AB	:FILESPEC			
		U.829: .BLKB	0		:
19F8	005AB	:TPASTYPE			
		U.896: .WORD	6648		:
0000*	005AD	:TPASSUBEXP			
		U.898: .WORD	<<U.897-U.898>-2>		:
0000*	005AF	:TPASTARGET			
		U.900: .WORD	<<U.899-U.900>-2>		:
15F6	005B1	:TPASTYPE			
		U.901: .WORD	5622		:
0000*	005B3	:TPASTARGET			

		U.902: .WORD	<<U.899-U.902>-2>	:
	005B5	:FIL0		
		U.899: .BLKB	0	
19F8	005B5	:TPASTYPE		
		U.903: .WORD	6648	:
0000*	005B7	:TPASSUBEXP		
		U.905: .WORD	<<U.904-U.905>-2>	:
0000*	005B9	:TPASTARGET		
		U.907: .WORD	<<U.906-U.907>-2>	:
15F6	005BB	:TPASTYPE		
		U.908: .WORD	5622	:
0000*	005BD	:TPASTARGET		
		U.909: .WORD	<<U.906-U.909>-2>	:
	005BF	:FIL1		
		U.906: .BLKB	0	
19F8	005BF	:TPASTYPE		
		U.910: .WORD	6648	:
0000*	005C1	:TPASSUBEXP		
		U.912: .WORD	<<U.911-U.912>-2>	:
0000*	005C3	:TPASTARGET		
		U.914: .WORD	<<U.913-U.914>-2>	:
15F6	005C5	:TPASTYPE		
		U.915: .WORD	5622	:
0000*	005C7	:TPASTARGET		
		U.916: .WORD	<<U.913-U.916>-2>	:
	005C9	:FIL2		
		U.913: .BLKB	0	
05F1	005C9	:TPASTYPE		
		U.917: .WORD	1521	:
19F8	005CB	:TPASTYPE		
		U.918: .WORD	6648	:
0000*	005CD	:TPASSUBEXP		
		U.920: .WORD	<<U.919-U.920>-2>	:
0000*	005CF	:TPASTARGET		
		U.922: .WORD	<<U.921-U.922>-2>	:
15F6	005D1	:TPASTYPE		
		U.923: .WORD	5622	:
0000*	005D3	:TPASTARGET		
		U.924: .WORD	<<U.921-U.924>-2>	:
	005D5	:FIL3		
		U.921: .BLKB	0	
19F8	005D5	:TPASTYPE		
		U.925: .WORD	6648	:
0000*	005D7	:TPASSUBEXP		
		U.927: .WORD	<<U.926-U.927>-2>	:
FFFF	005D9	:TPASTARGET		
		U.928: .WORD	-1	:
15F6	005DB	:TPASTYPE		
		U.929: .WORD	5622	:
FFFF	005DD	:TPASTARGET		
		U.930: .WORD	-1	:
	005DF	:NODESPEC		
		U.897: .BLKB	0	
1DF8	005DF	:TPASTYPE		
		U.931: .WORD	7672	:
0000*	005E1	:TPASSUBEXP		
		U.933: .WORD	<<U.932-U.933>-2>	:

143A	00611	;TPASTYPE	
		U.960: .WORD	5178
FFFF	00613	;TPASTARGET	
		U.961: .WORD	-1
	00615	;DIRECT	
		U.911: .BLKB	0
003C	00615	;TPASTYPE	
		U.962: .WORD	60
045B	00617	;TPASTYPE	
		U.963: .WORD	1115
19F8	00619	;TPASTYPE	
		U.964: .WORD	6648
0000*	0061B	;TPASSUBEXP	
		U.966: .WORD	<<U.965-U.966>-2>
0000*	0061D	;TPASTARGET	
		U.968: .WORD	<<U.967-U.968>-2>
0DF8	0061F	;TPASTYPE	
		U.969: .WORD	3576
0000*	00621	;TPASSUBEXP	
		U.971: .WORD	<<U.970-U.971>-2>
	00623	;DIR2	
		U.967: .BLKB	0
103E	00623	;TPASTYPE	
		U.972: .WORD	4158
FFFF	00625	;TPASTARGET	
		U.973: .WORD	-1
145D	00627	;TPASTYPE	
		U.974: .WORD	5213
FFFF	00629	;TPASTARGET	
		U.975: .WORD	-1
	0062B	;DIRSUB	
		U.970: .BLKB	0
11F1	0062B	;TPASTYPE	
		U.976: .WORD	4593
0000*	0062D	;TPASTARGET	
		U.978: .WORD	<<U.977-U.978>-2>
102D	0062F	;TPASTYPE	
		U.979: .WORD	4141
0000*	00631	;TPASTARGET	
		U.980: .WORD	<<U.977-U.980>-2>
05F6	00633	;TPASTYPE	
		U.981: .WORD	1526
	00635	;DIRSUB1	
		U.977: .BLKB	0
102E	00635	;TPASTYPE	
		U.982: .WORD	4142
0000*	00637	;TPASTARGET	
		U.983: .WORD	<<U.970-U.983>-2>
15F6	00639	;TPASTYPE	
		U.984: .WORD	5622
FFFF	0063B	;TPASTARGET	
		U.985: .WORD	-1
	0063D	;UFD	
		U.965: .BLKB	0
05F4	0063D	;TPASTYPE	
		U.986: .WORD	1524
042C	0063F	;TPASTYPE	

Page 52
(3)

Address	Offset	Field	Value
15F4	00641	U.987: .WORD	1068
		:TPASTYPE	
FFFF	00643	U.988: .WORD	5620
		:TPASTARGET	
	00645	U.989: .WORD	-1
		:TYPE	
042E	00645	U.919: .BLKB	0
		:TPASTYPE	
11F0	00647	U.990: .WORD	1070
		:TPASTYPE	
FFFF	00649	U.991: .WORD	4592
		:TPASTARGET	
11F2	0064B	U.992: .WORD	-1
		:TPASTYPE	
FFFF	0064D	U.993: .WORD	4594
		:TPASTARGET	
15F6	0064F	U.994: .WORD	-1
		:TPASTYPE	
FFFF	00651	U.995: .WORD	5622
		:TPASTARGET	
	00653	U.996: .WORD	-1
		:VERSION	
043B	00653	U.926: .BLKB	0
		:TPASTYPE	
11F3	00655	U.997: .WORD	1083
		:TPASTYPE	
FFFF	00657	U.998: .WORD	4595
		:TPASTARGET	
11F2	00659	U.999: .WORD	-1
		:TPASTYPE	
FFFF	0065B	U.1000: .WORD	4594
		:TPASTARGET	
15F6	0065D	U.1001: .WORD	-1
		:TPASTYPE	
FFFF	0065F	U.1002: .WORD	5622
		:TPASTARGET	
	00661	U.1003: .WORD	-1
		:PARSE_NUMBER_OR_NULL	
19F8	00661	U.142: .BLKB	0
		:TPASTYPE	
0000*	00663	U.1004: .WORD	6648
		:TPASSUBEXP	
FFFF	00665	U.1005: .WORD	<<U.87-U.1005>-2>
		:TPASTARGET	
15F6	00667	U.1006: .WORD	-1
		:TPASTYPE	
FFFF	00669	U.1007: .WORD	5622
		:TPASTARGET	
	0066B	U.1008: .WORD	-1
		:PARSE_NUMBER	
11F3	0066B	U.87: .BLKB	0
		:TPASTYPE	
FFFF	0066D	U.1009: .WORD	4595
		:TPASTARGET	
0425	0066F	U.1010: .WORD	-1
		:TPASTYPE	
		U.1011: .WORD	1061

B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H

1044	00671	;TPASTYPE			
		U.1012: .WORD	4164		;
0000*	00673	;TPASTARGET			
		U.1014: .WORD	<<U.1013-U.1014>-2>		;
1058	00675	;TPASTYPE			
		U.1015: .WORD	4184		;
0000*	00677	;TPASTARGET			
		U.1017: .WORD	<<U.1016-U.1017>-2>		;
044F	00679	;TPASTYPE			
		U.1018: .WORD	1103		;
15F4	0067B	;TPASTYPE			
		U.1019: .WORD	5620		;
FFFF	0067D	;TPASTARGET			
		U.1020: .WORD	-1		;
	0067F	;HEXNUM			
		U.1016: .BLKB	0		
15F5	0067F	;TPASTYPE			
		U.1021: .WORD	5621		;
FFFF	00681	;TPASTARGET			
		U.1022: .WORD	-1		;
	00683	;DECNUM			
		U.1013: .BLKB	0		
15F3	00683	;TPASTYPE			
		U.1023: .WORD	5619		;
FFFF	00685	;TPASTARGET			
		U.1024: .WORD	-1		;
	00687	;INCLUDE_QUAL			
		U.851: .BLKB	0		
0535	00687	;TPASTYPE			
		U.1028: .WORD	1333		;
19F8	00689	;TPASTYPE			
		U.1030: .WORD	6648		;
0000*	0068B	;TPASSUBEXP			
		U.1032: .WORD	<<U.1031-U.1032>-2>		;
0000*	0068D	;TPASTARGET			
		U.1034: .WORD	<<U.1033-U.1034>-2>		;
0DF8	0068F	;TPASTYPE			
		U.1035: .WORD	3576		;
0000*	00691	;TPASSUBEXP			
		U.1036: .WORD	<<U.93-U.1036>-2>		;
	00693	;INCL1			
		U.1033: .BLKB	0		
1028	00693	;TPASTYPE			
		U.1037: .WORD	4136		;
0000*	00695	;TPASTARGET			
		U.1039: .WORD	<<U.1038-U.1039>-2>		;
59F8	00697	;TPASTYPE			
		U.1040: .WORD	23032		;
0000*	00699	;TPASSUBEXP			
		U.1041: .WORD	<<U.124-U.1041>-2>		;
00000000*	0069B	;TPASADDR			
		U.1042: .LONG	<<INCLUDE_DESC-U.1042>-4>		;
FFFF	0069F	;TPASTARGET			
		U.1043: .WORD	-1		;
0DF8	006A1	;TPASTYPE			
		U.1044: .WORD	3576		;
0000*	006A3	;TPASSUBEXP			

		U.1045: .WORD	<<U.93-U.1045>-2>	:
006A5		:INCLUDE_LIST		
		U.1038: .BLKB	0	
5DF8	006A5	:TPASTYPE		
		U.1046: .WORD	24056	:
0000*	006A7	:TPASSUBEXP		
		U.1048: .WORD	<<U.1047-U.1048>-2>	:
00000000*	006A9	:TPASADDR		
		U.1049: .LONG	<<INCLUDE_DESC-U.1049>-4>	:
0000*	006AD	:TPASTARGET		
		U.1051: .WORD	<<U.1050-U.1051>-2>	:
	006AF	:INCL2		
		U.1050: .BLKB	0	
1029	006AF	:TPASTYPE		
		U.1052: .WORD	4137	:
FFFF	006B1	:TPASTARGET		
		U.1053: .WORD	-1	:
0DF8	006B3	:TPASTYPE		
		U.1054: .WORD	3576	:
0000*	006B5	:TPASSUBEXP		
		U.1055: .WORD	<<U.93-U.1055>-2>	:
	006B7	:PARSE_INCLUDE_LIST		
		U.1047: .BLKB	0	
19F8	006B7	:TPASTYPE		
		U.1056: .WORD	6648	:
0000*	006B9	:TPASSUBEXP		
		U.1057: .WORD	<<U.124-U.1057>-2>	:
0000*	006BB	:TPASTARGET		
		U.1059: .WORD	<<U.1058-U.1059>-2>	:
0DF8	006BD	:TPASTYPE		
		U.1060: .WORD	3576	:
0000*	006BF	:TPASSUBEXP		
		U.1061: .WORD	<<U.93-U.1061>-2>	:
	006C1	:INCL3		
		U.1058: .BLKB	0	
102C	006C1	:TPASTYPE		
		U.1062: .WORD	4140	:
0000*	006C3	:TPASTARGET		
		U.1063: .WORD	<<U.1047-U.1063>-2>	:
15F6	006C5	:TPASTYPE		
		U.1064: .WORD	5622	:
FFFF	006C7	:TPASTARGET		
		U.1065: .WORD	-1	:
	006C9	:SHARE_QUAL		
		U.875: .BLKB	0	
8736	006C9	:TPASTYPE		
		U.1069: .WORD	-30922	:
01	006CB	:TPASFLAGS2		
		U.1070: .BYTE	1	:
00000000	006CC	:TPASPARAM		
		U.1071: .LONG	0	:
00000000V	006D0	:TPASACTION		
		U.1072: .LONG	<<SET_SHRCOPYFLAG-U.1072>-4>	:
19F8	006D4	:TPASTYPE		
		U.1074: .WORD	6648	:
0000*	006D6	:TPASSUBEXP		
		U.1075: .WORD	<<U.1031-U.1075>-2>	:

0000*	006D8	:TPASTARGET			
		U.1077: .WORD	<<U.1076-U.1077>-2>		:
15F6	006DA	:TPASTYPE			:
		U.1078: .WORD	5622		:
FFFF	006DC	:TPASTARGET			:
		U.1079: .WORD	-1		:
	006DE	:PARSHAREVAL			:
		U.1076: .BLKB	0		:
9337	006DE	:TPASTYPE			:
		U.1083: .WORD	-27849		:
01	006E0	:TPASFLAGS2			:
		U.1084: .BYTE	1		:
00000000	006E1	:TPASPARAM			:
		U.1085: .LONG	0		:
00000000V	006E5	:TPASACTION			:
		U.1086: .LONG	<<SET_SHRCOPYFLAG-U.1086>-4>		:
FFFF	006E9	:TPASTARGET			:
		U.1087: .WORD	-1		:
0DF8	006EB	:TPASTYPE			:
		U.1088: .WORD	3576		:
0000*	006ED	:TPASSUBEXP			:
		U.1089: .WORD	<<U.93-U.1089>-2>		:
	006EF	:GET SYMBOL			:
		U.124: .BLKB	0		:
102E	006EF	:TPASTYPE			:
		U.1091: .WORD	4142		:
0000*	006F1	:TPASTARGET			:
		U.1093: .WORD	<<U.1092-U.1093>-2>		:
11F1	006F3	:TPASTYPE			:
		U.1094: .WORD	4593		:
0000*	006F5	:TPASTARGET			:
		U.1095: .WORD	<<U.1092-U.1095>-2>		:
15F6	006F7	:TPASTYPE			:
		U.1096: .WORD	5622		:
FFFE	006F9	:TPASTARGET			:
		U.1097: .WORD	-2		:
	006FB	:GETSYM1			:
		U.1092: .BLKB	0		:
102E	006FB	:TPASTYPE			:
		U.1098: .WORD	4142		:
0000*	006FD	:TPASTARGET			:
		U.1099: .WORD	<<U.1092-U.1099>-2>		:
11F1	006FF	:TPASTYPE			:
		U.1100: .WORD	4593		:
0000*	00701	:TPASTARGET			:
		U.1101: .WORD	<<U.1092-U.1101>-2>		:
15F6	00703	:TPASTYPE			:
		U.1102: .WORD	5622		:
FFFF	00705	:TPASTARGET			:
		U.1103: .WORD	-1		:
	00707	:QUALVALSEP			:
		U.1031: .BLKB	0		:
103A	00707	:TPASTYPE			:
		U.1104: .WORD	4154		:
FFFF	00709	:TPASTARGET			:
		U.1105: .WORD	-1		:
143D	0070B	:TPASTYPE			:


```

      FFFF 0070D :U.1106: .WORD 5181 ;
      :TPASTARGET ;
      0070F :U.1107: .WORD -1 ;
      :ERRORPARSE 1 ;
      81F7 0070F :U.93: .BLRB 0 ;
      :TPASTYPE ;
      00000000V 00711 :U.1108: .WORD -32265 ;
      :TPASACTION ;
      81F1 00715 :U.1109: .LONG <<MISSINGARGERR-U.1109>-4> ;
      :TPASTYPE ;
      00000000V 00717 :U.1110: .WORD -32271 ;
      :TPASACTION ;
      81F0 0071B :U.1111: .LONG <<SYNTAXERR-U.1111>-4> ;
      :TPASTYPE ;
      00000000V 0071D :U.1112: .WORD -32272 ;
      :TPASACTION ;
      81ED 00721 :U.1113: .LONG <<SYNTAXERR-U.1113>-4> ;
      :TPASTYPE ;
      00000000V 00723 :U.1114: .WORD -32275 ;
      :TPASACTION ;
      85F6 00727 :U.1115: .LONG <<SYNTAXERR-U.1115>-4> ;
      :TPASTYPE ;
      00000000V 00729 :U.1116: .WORD -31242 ;
      :TPASACTION ;
      0072D :U.1117: .LONG <<SYNTAXERR-U.1117>-4> ;
      :ENDLINE 1 ;
      11F7 0072D :U.96: .BLKB 0 ;
      :TPASTYPE ;
      FFFF 0072F :U.1118: .WORD 4599 ;
      :TPASTARGET ;
      81F1 00731 :U.1119: .WORD -1 ;
      :TPASTYPE ;
      00000000V 00733 :U.1120: .WORD -32271 ;
      :TPASACTION ;
      81F0 00737 :U.1121: .LONG <<SYNTAXERR-U.1121>-4> ;
      :TPASTYPE ;
      00000000V 00739 :U.1122: .WORD -32272 ;
      :TPASACTION ;
      85ED 0073D :U.1123: .LONG <<SYNTAXERR-U.1123>-4> ;
      :TPASTYPE ;
      00000000V 0073F :U.1124: .WORD -31251 ;
      :TPASACTION ;
      :U.1125: .LONG <<SYNTAXERR-U.1125>-4> ;

      .PSECT _LIB$KEY0$,NOWRT, SHR, PIC,1

      00000 KEY_TABLE::
      00000 :TPASKEY0 .BLKB 0
      0000* 00000 :U.1: .BLKB 0
      0000* 00000 :TPASKEY :
      0000* 00002 :U.79: .WORD <U.78-U.1> ;
      :TPASKEY :
      0000* 00004 :U.100: .WORD <U.99-U.1> ;
      :TPASKEY :
      0000* 00006 :U.118: .WORD <U.117-U.1> ;
      :TPASKEY :

```


0000*	00008	U.191: .WORD	<U.190-U.1>
		:TPASKEY	
0000*	0000A	U.232: .WORD	<U.231-U.1>
		:TPASKEY	
0000*	0000C	U.250: .WORD	<U.249-U.1>
		:TPASKEY	
0000*	0000E	U.290: .WORD	<U.289-U.1>
		:TPASKEY	
0000*	00010	U.298: .WORD	<U.297-U.1>
		:TPASKEY	
0000*	00012	U.306: .WORD	<U.305-U.1>
		:TPASKEY	
0000*	00014	U.317: .WORD	<U.316-U.1>
		:TPASKEY	
0000*	00016	U.333: .WORD	<U.332-U.1>
		:TPASKEY	
0000*	00018	U.342: .WORD	<U.341-U.1>
		:TPASKEY	
0000*	0001A	U.360: .WORD	<U.359-U.1>
		:TPASKEY	
0000*	0001C	U.376: .WORD	<U.375-U.1>
		:TPASKEY	
0000*	0001E	U.396: .WORD	<U.395-U.1>
		:TPASKEY	
0000*	00020	U.404: .WORD	<U.403-U.1>
		:TPASKEY	
0000*	00022	U.415: .WORD	<U.414-U.1>
		:TPASKEY	
0000*	00024	U.431: .WORD	<U.430-U.1>
		:TPASKEY	
0000*	00026	U.437: .WORD	<U.436-U.1>
		:TPASKEY	
0000*	00028	U.446: .WORD	<U.445-U.1>
		:TPASKEY	
0000*	0002A	U.460: .WORD	<U.459-U.1>
		:TPASKEY	
0000*	0002C	U.489: .WORD	<U.488-U.1>
		:TPASKEY	
0000*	0002E	U.497: .WORD	<U.496-U.1>
		:TPASKEY	
0000*	00030	U.505: .WORD	<U.504-U.1>
		:TPASKEY	
0000*	00032	U.513: .WORD	<U.512-U.1>
		:TPASKEY	
0000*	00034	U.521: .WORD	<U.520-U.1>
		:TPASKEY	
0000*	00036	U.529: .WORD	<U.528-U.1>
		:TPASKEY	
0000*	00038	U.537: .WORD	<U.536-U.1>
		:TPASKEY	
0000*	0003A	U.545: .WORD	<U.544-U.1>
		:TPASKEY	
0000*	0003C	U.553: .WORD	<U.552-U.1>
		:TPASKEY	
0000*	0003E	U.561: .WORD	<U.560-U.1>
		:TPASKEY	
		U.569: .WORD	<U.568-U.1>


```

0000* 00040 :TPASKEY
              U.577: .WORD <U.576-U.1> ;
0000* 00042 :TPASKEY
              U.585: .WORD <U.584-U.1> ;
0000* 00044 :TPASKEY
              U.593: .WORD <U.592-U.1> ;
0000* 00046 :TPASKEY
              U.601: .WORD <U.600-U.1> ;
0000* 00048 :TPASKEY
              U.609: .WORD <U.608-U.1> ;
0000* 0004A :TPASKEY
              U.617: .WORD <U.616-U.1> ;
0000* 0004C :TPASKEY
              U.625: .WORD <U.624-U.1> ;
0000* 0004E :TPASKEY
              U.633: .WORD <U.632-U.1> ;
0000* 00050 :TPASKEY
              U.641: .WORD <U.640-U.1> ;
0000* 00052 :TPASKEY
              U.649: .WORD <U.648-U.1> ;
0000* 00054 :TPASKEY
              U.657: .WORD <U.656-U.1> ;
0000* 00056 :TPASKEY
              U.665: .WORD <U.664-U.1> ;
0000* 00058 :TPASKEY
              U.673: .WORD <U.672-U.1> ;
0000* 0005A :TPASKEY
              U.681: .WORD <U.680-U.1> ;
0000* 0005C :TPASKEY
              U.717: .WORD <U.716-U.1> ;
0000* 0005E :TPASKEY
              U.733: .WORD <U.732-U.1> ;
0000* 00060 :TPASKEY
              U.749: .WORD <U.748-U.1> ;
0000* 00062 :TPASKEY
              U.777: .WORD <U.776-U.1> ;
0000* 00064 :TPASKEY
              U.795: .WORD <U.794-U.1> ;
0000* 00066 :TPASKEY
              U.858: .WORD <U.857-U.1> ;
0000* 00068 :TPASKEY
              U.866: .WORD <U.865-U.1> ;
0000* 0006A :TPASKEY
              U.1026: .WORD <U.1025-U.1> ;
0000* 0006C :TPASKEY
              U.1067: .WORD <U.1066-U.1> ;
0000* 0006E :TPASKEY
              U.1081: .WORD <U.1080-U.1> ;

```

.PSECT \$SPLIT\$,NOWRT,NOEXE,2

```

63 69 66 69 74 6E 65 64 69 20 65 67 61 6D 14 00000 P.AAA: .BYTE 20 ;
6E 6F 69 74 49 00001 .ASCII \Image identification\ ;
61 00010 ;
0A 00015 P.AAB: .BYTE 10 ;
65 6D 61 6E 20 65 67 61 6D 49 00016 .ASCII \Image name\ ;
07 00020 P.AAC: .BYTE 7 ;

```


LNK_PROCOPTIONS Linker options parser
V04=000 readoption - Read record from options file

H 16
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 59
(3)

73	75	6C	63	20	74	6C	75	61	66	20	65	67	61	50	00021		.ASCII	\Channel\	:
												72	65	74	00028	P.AAD:	.BYTE	18	:
															00029		.ASCII	\Page fault cluster\	:
3E	6E	6F	69	74	70	6F	20	72	65	6B	6E	69	4C	3C	00038				:
															0003B	P.AAE:	.BYTE	15	:
															0003C		.ASCII	\<Linker option>\	:
				45	4C	49	46	20	4E	4F	49	54	50	4F	0004B	P.AAF:	.BYTE	11	:
															0004C		.ASCII	\OPTION FILE\	:
															00057	P.AAG:	.BYTE	8	:
															00058		.ASCII	\DZRO_MIN\	:
															00060	P.AAH:	.BYTE	7	:
															00061		.ASCII	\GSMATCH\	:
															00068	P.AAI:	.BYTE	5	:
															00069		.ASCII	\IOSEG\	:
															0006E	P.AAJ:	.BYTE	7	:
															0006F		.ASCII	\ISD_MAX\	:
															00076	P.AAK:	.BYTE	11	:
															00077		.ASCII	\Unsupported\	:
															00082	P.AAL:	.BYTE	5	:
															00083		.ASCII	\Stack\	:

.PSECT \$OWN\$,NOEXE,2

00000008	00000	TPARSE_BLOCK:	
		.LONG	8
00000004	00004	.LONG	4
	00008	.BLKB	28
	00024	CMDBUFFER:	
		.BLKB	4
	00028	OPTLINEDESC:	
		.BLKB	8
	00030	FILEFLAGSADR:	
		.BLKB	4
	00034	OPTRABADR:	
		.BLKB	4
	00038	INCLUDE_DESC:	
		.BLKB	8
	00040	CURCOLLECTDESC:	
		.BLKB	4
	00044	CURPSECTDESC:	
		.BLKB	4
	00048	CURSYMDESC:	
		.BLKB	4
	0004C	CURSYMSNB:	
		.BLKB	4
	00050	FILESPEC_DESC:	
		.BLKB	8
	00058	OPTIONFILENAME:	
		.BLKB	4
	0005C	QUOTEFLAG:	
		.BLKB	4
	00060	CLUOPTFLAG:	
		.BLKB	4
	00064	SHARECOPY:	
		.BLKB	4
	00068	PROTECTFLAG:	
		.BLKB	4


```

.PSECT $GLOBALS,NOEXE,2
00000 LNK$GL_INCLST::
        .BLKB 4
00004 LNK$GL_DEFCLUNUM::
        .BLKB 4
00008 LNK$GL_OPTEXTP::
        .BLKB 4
00000000' 0000C LNK$GL_OPTEXTE::
        .ADDRESS LNK$GL_OPTEXTP
00010 LNK$GL_SHLEXTRA::
        .BLKB 4
00014 LNK$GW_CHANS::
        .BLKB 2
00016 LNK$GW_IOSEG::
        .BLKB 2
FFFFFFFF FFFFFFFF 00018 LNK$GQ_PRIVS::
        .LONG -1, -1
00000000 00020 LNK$GL_UNSUPPORTED::
        .LONG 0
00024 LNK$GL_CCLULST::
        .BLKB 4
00028 LNK$GL_PSCDFLST::
        .BLKB 4

LNK$K_MAXCHANS== 4095
LNK$K_MAXSTACK== 65535
LNK$K_MAXIOSEG== 65535
LNK$K_MAXISDS== 65535
LNK$K_MINDZRO== 65535
LNK$K_MAXMAJID== 255
LNK$K_MAXMINID== 16777215
LNK$K_MAXOPTLIN== 32767
LNK$K_MAXPFC== 255
IMAGE_IDENT_NAME= P.AAA
IMAGE_NAME_NAME= P.AAB
CHANNELS_NAME= P.AAC
CLUSTERPFC_NAME= P.AAD
DEFINED_BY_OPTION= P.AAE
OPTIONFILESTRING= P.AAF
DZROMIN_NAME= P.AAG
GSMATCH_NAME= P.AAH
IOSEG_NAME= P.AAI
ISDMAX_NAME= P.AAJ
UNSUPPORTED_NAME= P.AAK
STACK_NAME= P.AAL

.EXTRN INPUTFILE, CRF$INSRTREF
.EXTRN CRF$INSRTKEY, LIB$INSERT_TREE
.EXTRN LIB$LOOKUP_TREE
.EXTRN LNK$ALLOBLR, LNK$ALLOC_PDD
.EXTRN LNK$ALLOCLUSTER
.EXTRN LNK$COMPARE_PDD
.EXTRN LNK$CLUNAMCMP, LNK$INSERT_CLU
.EXTRN LNK$DEALBLK, LNK$INSUDFSYM
.EXTRN LNK$INSERT, LNK$SEARCH
.EXTRN LNK$SETLIBRIN, LNK$SETSHRBLIN

```



```
.EXTRN LNK$UPCASE_D, LIB$PUT_OUTPUT
.EXTRN LIB$TPARSE, SYSS$FAO
.EXTRN LNK$GB_MATCHCTL
.EXTRN LNK$GL_MATCHID, LNK$GL_DEFCLU
.EXTRN LNK$GL_CURCLU, LNK$GL_CLUTREE
.EXTRN LNK$GL_CTLMSK, LNK$GL_FVMLST
.EXTRN LNK$GW_STACK, LNK$GT_PSCSTRING
.EXTRN LNK$GT_CLUSTRING
.EXTRN LNK$GT_SYMSTRING
.EXTRN LNK$GT_IMGNAM, LNK$GT_IMGID
.EXTRN LNK$AL_VALCTLTB
.EXTRN LNK$AL_SYTBLEMT
.EXTRN LNK$GL_MAXSYMSZ
.EXTRN LNK$GL_MAXMODSZ
.EXTRN LNK$GL_SYSINPUT_FLAG
.EXTRN LNK$GW_NCROSRFS
.EXTRN LNK$GW_MISECTS, LNK$GW_DZROMIN
.EXTRN LNK$GW_NUDFSYMS
.EXTRN LNK$CMDTOOLONG
.EXTRN LNK$CONFQUAL, LNK$CRFERR
.EXTRN LNK$ILLNAMELEN
.EXTRN LNK$LINERR, LNK$MULCLUOPT
.EXTRN LNK$OPTIGNSHR, LNK$OPTIGNSYS
.EXTRN LNK$OPTLIN, LNK$OPTSYNERR
.EXTRN LNK$OPTVALERR, LNK$OPTARGMIS
.EXTRN LNK$PREMEOF, LNK$READERR
.EXTRN LNK$SHRCPYIGN, LNK$SHRINSYS
.EXTRN LNK$SHRSEPCLU, LNK$R_MAX_FILENAME_LENGTH
.EXTRN SYSS$GET
```

.PSECT \$CODE\$,NOWRT,2

01FC 00000 READOPTION:

	58	00000000'	EF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	0998
	5E		04	C2	00009	MOVAB	OPTRABADR, R8	
			68	DD	0000C	SUBL2	#4, SP	
00000000G	00		01	FB	0000E	PUSHL	OPTRABADR	1016
	57		50	D0	00015	CALLS	#1, SYSS\$GET	
	53	04	AC	D0	00018	MOVL	RO, STATUS	
	52		68	D0	0001C	MOVL	LINEDESC, R3	1017
	63	22	A2	B0	0001F	MOVL	OPTRABADR, R2	
04	A3	28	A2	D0	00023	MOVW	34(R2), (R3)	
	48		57	E9	00028	MOVL	40(R2), 4(R3)	1018
						BLBC	STATUS, 2\$	1020
10	00000000G		04	E1	0002B	BBC	#4, LNK\$GL_CTLMSK+1, 1\$	1022
	09	00000000G	00	E8	00033	BLBS	LNK\$GL_SYSINPUT_FLAG, 1\$	1023
			53	DD	0003A	PUSHL	R3	1024
00000000G	00		01	FB	0003C	CALLS	#1, LIB\$PUT OUTPUT	
	50		63	3C	00043	MOVZWL	(R3), BLOCKSIZE	1026
	50		06	C0	00046	ADDL2	#6, BLOCKSIZE	
		4001	8F	BB	00049	PUSHR	#^M<R0,SP>	1027
00000000G	00		02	FB	0004D	CALLS	#2, LNK\$ALLOBLK	
	56		6E	D0	00054	MOVL	BLOCKADDR, R6	1028
			66	D4	00057	CLRL	(R6)	
	04	A6	63	B0	00059	MOVW	(R3), 4(R6)	1029
06	A6	04	63	28	0005D	MOVW	(R3), 4(R3), 6(R6)	1032
	00000000'	FF	56	D0	00063	MOVL	R6, @LNK\$GL_OPTEXTE	1033

LNK_PROCOPTIONS Linker options parser
V04=000 readoption - Read record from options file

K 16
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 62
(3)

00000000'	EF	56	D0	0006A	MOVL	R6, LNK\$GL_OPTEXTE	:	1034
		40	11	00071	BRB	4\$	:	1020
0001827A	8F	57	D1	00073	2\$:	CMPL	STATUS, #98938	:
		20	12	0007A	BNEQ	3\$	:	1040
	02	6C	91	0007C	CMPB	(AP), #2	:	1042
		32	1F	0007F	BLSSU	4\$	:	
		08	AC	D5 00081	TSTL	8(AP)	:	
			2D	13 00084	BEQL	4\$	:	1044
		0C	A2	DD 00086	PUSHL	12(R2)	:	
			57	DD 00089	PUSHL	STATUS	:	
		24	A8	DD 0008B	PUSHL	OPTIONFILENAME	:	1043
			01	DD 0008E	PUSHL	#1	:	
		08	AC	DD 00090	PUSHL	ERRORSIGNAL	:	
00000000G	00		05	FB 00093	CALLS	#5, LIB\$SIGNAL	:	1040
			17	11 0009A	BRB	4\$	:	1048
		0C	A2	DD 0009C	3\$:	PUSHL	12(R2)	:
			57	DD 0009F	PUSHL	STATUS	:	
		24	A8	DD 000A1	PUSHL	OPTIONFILENAME	:	1047
			01	DD 000A4	PUSHL	#1	:	
		00000000G	8F	DD 000A6	PUSHL	#LINS_READERR	:	
00000000G	00		05	FB 000AC	CALLS	#5, LIB\$STOP	:	
	50		57	D0 000B3	4\$:	MOVL	STATUS, R0	:
			04	000B6	RET		:	1052
							:	1053

; Routine Size: 183 bytes, Routine Base: \$CODE\$ + 0000

; 942 1054 1


```

: 944      1055 1 %sbttl 'LNK$PROCOPTNS -- Entry point to options parsing';
: 945      1056 1 global routine lnk$procoptns ( optfilename, fileflags, optionrab) =
: 946      1057 2   begin
: 947      1058 2   |
: 948      1059 2   | This routine parses the options file
: 949      1060 2   |
: 950      1061 2   | Inputs:
: 951      1062 2   |
: 952      1063 2   |     optfilename      The address of the string descriptor for the options file
: 953      1064 2   |     fileflags        Address of file control flags
: 954      1065 2   |     optionrab         Address of the RAB connected for GET sequential
: 955      1066 2   |
: 956      1067 2   | Outputs:
: 957      1068 2   |
: 958      1069 2   |     Linker data structures created according to inputs found in options file.
: 959      1070 2   |
: 960      1071 2   | --
: 961      1072 2   | map
: 962      1073 2   |     optfilename : ref bblock,
: 963      1074 2   |     fileflags   : ref bblock,
: 964      1075 2   |     optionrab   : ref bblock;
: 965      1076 2   | local
: 966      1077 2   |     blockaddr  : ref bblock,
: 967      1078 2   |     blocksize,
: 968      1079 2   |     linedesc   : bblock [dsc$sc_s_bln],
: 969      1080 2   |     commentline,
: 970      1081 2   |     rmsstatus;
: 971      1082 2   |     fileflagsadr = .fileflags;           !Put file flags address into own storage
: 972      1083 2   |     optrabadr   = .optionrab;           !Put RAB address in own storage
: 973      1084 2   |     optionfilename = .optfilename;       !Put string descriptor address into own storage
: 974      1085 2   |     ch$fill (0, dsc$sc_s_bln, linedesc); !Clear out the descriptor
: 975      1086 2   |
: 976      1087 2   | Copy option file name out to the option text list
: 977      1088 2   |
: 978      1089 2   |
: 979      1090 2   | if .lnk$gl_ctlmsk [lnk$v_long]           !If creating a long map
: 980      1091 2   | then
: 981      1092 2   |     begin
: 982      1093 2   |         blocksize = oeb$sc_size + .optionfilename [dsc$w_length] + 2; !Figure size of block to allocate
: 983      1094 2   |         lnk$alloblk (.blocksize, blockaddr); !Allocate it
: 984      1095 2   |         blockaddr [oeb$l_nxttoeb] = 0;
: 985      1096 2   |         blockaddr [oeb$w_bytcnt] = .optionfilename [dsc$w_length] + 2; !Set size of line into descriptor
: 986      1097 2   |         begin
: 987      1098 2   |             bind
: 988      1099 2   |                 filenamestring = blockaddr [oeb$t_text] : vector [, byte];
: 989      1100 2   |
: 990      1101 2   |                 filenamestring [0] = %ascii'<';
: 991      1102 2   |                 ch$move (.optionfilename [dsc$w_length], !Copy the filename out
: 992      1103 2   |                     .optionfilename [dsc$a_pointer], filenamestring [1]);
: 993      1104 2   |                 filenamestring [.optionfilename [dsc$w_length] + 1] = %ascii'>';
: 994      1105 2   |             end;
: 995      1106 2   |             lnk$gl_optexte [oeb$l_nxttoeb] = .blockaddr; !Link the block into the end of the list
: 996      1107 2   |             lnk$gl_optexte = .blockaddr; ! and make it the last one
: 997      1108 2   |         end;
: 998      1109 2   |
: 999      1110 2   |
: 1000     1111 2   | Allocate a buffer. This buffer will be used to pack continuation lines into one continuous line
```



```

1001      1112 2 !      lnk$alloblk (lnk$k_maxoptlin, cmdbuffer);
1002      1113 2 !
1003      1114 2 !
1004      1115 2 ! Loop, reading and parsing commands until end of file
1005      1116 2 !
1006      1117 2 ! while (rmsstatus = readoption (linedesc)) neq rms$_eof do
1007      1118 3 ! begin
1008      1119 3 !     tparse_block [tpa$l_stringcnt] = 0;           !Zero line length
1009      1120 3 !     tparse_block [tpa$l_stringptr] = .cmdbuffer;   !Point TPARSE to the buffer
1010      1121 3 !     quoteflag = false;                               !Not inside double quotes
1011      1122 3 !     cluoptflag = false;                             !Not processing cluster option
1012      1123 3 !
1013      1124 3 !     if .linedesc [dsc$w_length] neq 0             !If line is not null
1014      1125 3 !     then
1015      1126 3 !         scanline (linedesc, commentline);         ! scan the line and store in the buffer
1016      1127 3 !
1017      1128 3 !     if .tparse_block [tpa$l_stringcnt] neq 0
1018      1129 3 !     then
1019      1130 4 !         begin
1020      1131 4 !             ch$move (dsc$c_s_bln, tparse_block [tpa$l_stringcnt],
1021      1132 4 !                 !Make a copy of the options line descriptor
1022      1133 4 !                 optlinedesc);
1023      1134 4 !             lnk$upcase_d (tparse_block [tpa$l_stringcnt]);      !Convert string to upper case
1024      1135 4 !
1025      1136 4 !         Parse the command line
1026      1137 4 !
1027      1138 4 !             if not lib$tparse (tparse_block, state_table, key_table) then syntaxerr ();
1028      1139 4 !
1029      1140 4 !         Reset current cluster to default cluster at end of each options line
1030      1141 4 !
1031      1142 4 !             lnk$gl_curclu = lnk$gl_defclu;
1032      1143 3 !         end;
1033      1144 2 !     end;
1034      1145 2 !
1035      1146 2 ! Deallocate the buffer
1036      1147 2 !
1037      1148 2 !     lnk$dealblk (lnk$k_maxoptlin, .cmdbuffer);
1038      1149 2 !     return true
1039      1150 1 ! end;

```

08		00		28		50		07FC 00000		.ENTRY		LNK\$PROCOPTS, Save R2,R3,R4,R5,R6,R7,R8,-		1056	
				5A	00000000G	00	9E	00002		MOVAB	R9,R10				
				59	00000000'	EF	9E	00009		MOVAB	LNK\$ALLOBLK, R10				
				5E		10	C2	00010		SUBL2	TPARSE BLOCK+8, R9				
				A9	08	AC	7D	00013		MOVQ	#16, SP				
				A9	04	AC	D0	00018		MOVL	FILEFLAGS, FILEFLAGSADR				1082
				6E		00	2C	0001D		MOVLC	OPTIONFILENAME, OPTIONFILENAME				1084
					08	AE		00022		MOVCL	#0, (SP), #0, #8, LINEDESC				1085
				3F	00000000G	00	E9	00024		BLBC	LNK\$GL CTLMSK+1, 1\$				1090
				50	50	B9	3C	0002B		MOVZWL	OPTIONFILENAME, BLOCKSIZE				1093
				50		08	C0	0002F		ADDL2	#8, BLOCKSIZE				
					4001	8F	BB	00032		PUSHR	#*M<R0,SP>				1094

LNK_PROCOPTIONS
V04=000

Linker options parser
LNK\$PROCOPTNS -- Entry point to options parsing

B 1
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 65
(4)

04	A7	58	50	02	FB	00036	CALLS	#2, LNK\$ALLOBLK	:	1095
		57		6E	D0	00039	MOVL	BLOCKADDR, R7	:	
		56	06	67	D4	0003C	CLRL	(R7)	:	
		68		A9	D0	C003E	MOVL	OPTIONFILENAME, R8	:	1096
		56		02	A1	00042	ADDW3	#2, (R8), 4(R7)	:	
		66		A7	9E	00047	MOVAB	6(R7), R6	:	1099
01	A6	04		3C	90	0004B	MOVB	#60, (R6)	:	1101
		50		68	28	0004E	MOV3	(R8), @4(R8), 1(R6)	:	1103
	01	A046		68	3C	00054	MOVZWL	(R8), R0	:	1104
	00000000'	FF		3E	90	00057	MOVB	#62, 1(R0)[R6]	:	
	00000000'	EF		57	D0	0005C	MOVL	R7, @LNK\$GL_OPTEXTE	:	1106
			1C	57	D0	00063	MOVL	R7, LNK\$GL_OPTEXTE	:	1107
		7E	7FFF	A9	9F	0006A	PUSHAB	CMDBUFFER	:	1113
		6A		8F	3C	0006D	MOVZWL	#32767, -(SP)	:	
			08	02	FB	00072	CALLS	#2, LNK\$ALLOBLK	:	
	FECC	CF		AE	9F	00075	PUSHAB	LINEDESC	:	1117
		56		01	FB	00078	CALLS	#1, READOPTION	:	
	0001827A	8F		50	D0	0007D	MOVL	R0, RMSSTATUS	:	
				56	D1	00080	CMPL	RMSSTATUS, #98938	:	
				5B	13	00087	BEQL	5\$	:	
		04	A9	69	D4	00089	CLRL	TPARSE_BLOCK+8	:	1119
			1C	A9	D0	0008B	MOVL	CMDBUFFER, TPARSE_BLOCK+12	:	1120
			54	A9	7C	00090	CLRL	QUOTEFLAG	:	1121
			08	AE	B5	00093	TSTW	LINEDESC	:	1124
				0D	13	00096	BEQL	3\$	:	
			04	AE	9F	00098	PUSHAB	COMMENTLINE	:	1126
			0C	AE	9F	0009B	PUSHAB	LINEDESC	:	
	00000000V	EF		02	FB	0009E	CALLS	#2, SCANLINE	:	
				69	D5	000A5	TSTL	TPARSE_BLOCK+8	:	1128
				CC	13	000A7	BEQL	2\$	:	
20	A9	69		08	28	000A9	MOV3	#8, TPARSE_BLOCK+8, OPTLINEDESC	:	1131
				59	DD	000AE	PUSHL	R9	:	1134
	00000000G	00		01	FB	000B0	CALLS	#1, LNK\$UPCASE_D	:	
			00000000'	EF	9F	000B7	PUSHAB	KEY TABLE	:	1138
			00000000'	EF	9F	000BD	PUSHAB	STATE TABLE	:	
			FB	A9	9F	000C3	PUSHAB	TPARSE_BLOCK	:	
	00000000G	00		03	FB	000C6	CALLS	#3, LIB\$TPARSE	:	
		07		50	E8	000CD	BLBS	R0, 4\$	:	
	00000000V	EF		00	FB	000D0	CALLS	#0, SYNTAXERR	:	
	00000000G	00	00000000G	00	9E	000D7	MOVAB	LNK\$GL_DEFCLU, LNK\$GL_CURCLU	:	1142
				91	11	000E2	BRB	2\$	:	1117
			1C	A9	DD	000E4	PUSHL	CMDBUFFER	:	1148
		7E	7FFF	8F	3C	000E7	MOVZWL	#32767, -(SP)	:	
	00000000G	00		02	FB	000EC	CALLS	#2, LNK\$DEALBLK	:	
		50		01	D0	000F3	MOVL	#1, R0	:	1149
				04	000F6		RET		:	1150

; Routine Size: 247 bytes, Routine Base: \$CODE\$ + 00B7


```
: 1041      1151 1 %sbtll 'Form complete options line, handling continuations';
: 1042      1152 1 routine scanline (indesc, commentline) =
: 1043      1153 2   begin
: 1044      1154 2   |
: 1045      1155 2   | This routine scans a line and handles continuations.
: 1046      1156 2   |
: 1047      1157 2   | Inputs:
: 1048      1158 2   |
: 1049      1159 2   |     indesc  address of string descriptor for input line
: 1050      1160 2   |
: 1051      1161 2   | Outputs:
: 1052      1162 2   |
: 1053      1163 2   |     tparse_block[tpa$l_stringcnt] set to count of characters in .cmdbuffer
: 1054      1164 2   |     commentline      set true if line was a comment line (used in recursive
: 1055      1165 2   |                     internal calls only)
: 1056      1166 2   |
: 1057      1167 2   |     Blanks and tabs are stripped from line, continuation lines processed.
: 1058      1168 2   | ---
: 1059      1169 2   |
: 1060      1170 2   | Routine to store a character into the buffer
: 1061      1171 2   |
: 1062      1172 2   | routine storecharacter (cchar) =
: 1063      1173 2   |   begin
: 1064      1174 2   |     if .tparse_block [tpa$l_stringcnt] eql lnk$k_maxoptlin - 1
: 1065      1175 2   |     then
: 1066      1176 2   |       signal_stop (lin$_cmdtoolong, 1,
: 1067      1177 2   |         .optionfilename);
: 1068      1178 2   |
: 1069      1179 2   |     if (.cchar neq %c' ' and .cchar neq %c' ')      !If character is not space or tab
: 1070      1180 2   |       or .quoteflag                                ! or we are inside double quotes
: 1071      1181 2   |     then
: 1072      1182 2   |       begin
: 1073      1183 2   |         bind
: 1074      1184 2   |           charvector = .tparse_block [tpa$l_stringptr] + .tparse_block [tpa$l_stringcnt] : vector [,
: 1075      1185 2   |             byte];
: 1076      1186 2   |
: 1077      1187 2   |           charvector [0] = .cchar;                !Store the character
: 1078      1188 2   |           tparse_block [tpa$l_stringcnt] = .tparse_block [tpa$l_stringcnt] + 1;
: 1079      1189 2   |
: 1080      1190 2   |           if .cchar eql %c'"'                    !If double quote, then toggle the quote flag
: 1081      1191 2   |           then
: 1082      1192 2   |             quoteflag = not .quoteflag;
: 1083      1193 2   |           end;
: 1084      1194 2   |       return true
: 1085      1195 2   |     end;
:                                     !of storecharacter
```

0004 00000 STORECHARACTER:

					.WORD	Save R2	: 1172
					MOVAB	TPARSE_BLOCK+8, R2	: 1174
00007FFE	52	00000000'	EF	9E	00002		
	8F		62	D1	00009		
			12	12	00010		
		50	A2	DD	00012		: 1177
			01	DD	00015		: 1176
					PUSHL	OPTIONFILENAME	
					PUSHL	#1	

00000000G	00	00000000G	8F	DD	00017	PUSHL	#LINS_CMDTOOLONG	:	
	20		03	FB	0001D	CALLS	#3, LIB\$STOP	:	
		04	AC	D1	00024	1\$:	CMPL	CCHAR, #32	1179
	09		06	13	00028	BEQL	2\$	:	
		04	AC	D1	0002A	CMPL	CCHAR, #9	:	
			04	12	0002E	BNEQ	3\$	:	
50	16	54	A2	E9	00030	2\$:	BLBC	QUOTEFLAG, 4\$	1180
	A2		62	C1	00034	3\$:	ADDL3	TPARSE_BLOCK+8, TPARSE_BLOCK+12, R0	1184
	60	04	AC	90	00039	MOVB	CCHAR, (R0)	:	1187
			62	D6	0003D	INCL	TPARSE_BLOCK+8	:	1188
	22	04	AC	D1	0003F	CMPL	CCHAR, #34	:	1190
			05	12	00043	BNEQ	4\$	:	
	54	A2	54	A2	D2	00045	MCOML	QUOTEFLAG, QUOTEFLAG	1192
	50		01	D0	0004A	4\$:	MOVL	#1, R0	1194
			04	0004D	RET			:	1195

: Routine Size: 78 bytes, Routine Base: \$CODE\$ + 01AE

```
1086 1196 2 |
1087 1197 2 | Main body of scanline
1088 1198 2 |
1089 1199 2 | map
1090 1200 2 |     indesc : ref bblock;
1091 1201 2 |     local
1092 1202 2 |         rmsstatus,
1093 1203 2 |         stringstart,
1094 1204 2 |         cchar : byte,
1095 1205 2 |         tempdesc : bblock [dsc$c_s_bln];
1096 1206 2 |
1097 1207 2 |     if .indesc [dsc$w_length] eql 0 then return (.commentline = true);
1098 1208 2 |
1099 1209 2 |     ch$fill (0, dsc$c_s_bln, tempdesc); ! Clear descriptor
1100 1210 2 |     stringstart = .tparse_block [tpa$l_stringcnt];
1101 1211 2 |
1102 1212 2 | Copy the line into the command buffer a character at a time, watching for
1103 1213 2 | continuation lines and comments.
1104 1214 2 |
1105 1215 2 |     incru i from 0 to .indesc [dsc$w_length] - 1 do
1106 1216 2 |         begin
1107 1217 3 |             bind
1108 1218 3 |                 linebuffer = .indesc [dsc$a_pointer] : vector [, byte];
1109 1219 3 |
1110 1220 3 |                 if (cchar = .linebuffer [.i]) eql %ascii'!' !If character is comment character
1111 1221 3 |                 then
1112 1222 3 |                     exitloop (.commentline = true); ! then done with line
1113 1223 3 |                 if .cchar eql %ascii'-' !If character is a possible continuation character
1114 1224 3 |                 then
1115 1225 4 |                     begin ! then check it out
1116 1226 4 |                         if .i eql .indesc [dsc$w_length] - 1 !Are we on the last character?
1117 1227 4 |                         or
1118 1228 5 |                         begin ! or is everything following it blank or a comment?
1119 1229 5 |                             local
1120 1230 5 |                             status;
1121 1231 5 |                             status = true;
1122 1232 5 |                             incru j from .i + 1 to .indesc [dsc$w_length] - 1 do
1123 1233 5 |                                 if .linebuffer [.j] eql %ascii'!' !It's a continuation if comment follows
```



```

: 1124      1234 5      then
: 1125      1235 6      exitloop (.commentline = status = true)
: 1126      1236 5      else
: 1127      1237 5      if .linebuffer [.j] neq %ascii' '      ! but not if it's not a space
: 1128      1238 5      and .linebuffer [.j] neq %ascii'      ! or tab
: 1129      1239 5      then
: 1130      1240 5      exitloop (.commentline = status = false);
: 1131      1241 5      .status
: 1132      1242 5      end
: 1133      1243 5      : It is a continuation line
: 1134      1244 5      :
: 1135      1245 5      then
: 1136      1246 4      while true do
: 1137      1247 4      begin
: 1138      1248 5      rmsstatus = readoption (tempdesc, lin$_premeof);      !Read next line
: 1139      1249 5      scanline (tempdesc, .commentline);      !Scan the line
: 1140      1250 5
: 1141      1251 5      if not ..commentline      !If we did not just
: 1142      1252 5      then
: 1143      1253 5      return true;      ! scan a comment then done
: 1144      1254 5
: 1145      1255 5      end
: 1146      1256 5      : Not a continuation line
: 1147      1257 5
: 1148      1258 5      else
: 1149      1259 4      storecharacter (.cchar);
: 1150      1260 4      end
: 1151      1261 4      else
: 1152      1262 3      storecharacter (.cchar);
: 1153      1263 3      end;
: 1154      1264 2      if ..commentline      !If we just scanned a comment line
: 1155      1265 2      and (.tparse_block [tpa$L_stringcnt] eql .stringstart) ! with nothing else on it
: 1156      1266 3      then
: 1157      1267 2      return (.commentline = true)
: 1158      1268 3      else
: 1159      1269 2      begin
: 1160      1270 3      .commentline = false;
: 1161      1271 3      return true
: 1162      1272 3      end;
: 1163      1273 2      end;
: 1164      1274 1      end;
```

07FC 00000 SCANLINE:

5A	00000000'	EF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	: 1152
59	FDF7	CF	9E	00009	MOVAB	TPARSE_BLOCK+8, R10	:
5E		08	C2	0000E	MOVAB	READOPTION, R9	:
	04	BC	B5	00011	SUBL2	#8, SP	:
		07	12	00014	TSTW	@INDESC	: 1207
-	08	BC	01	D0	BNEQ	2\$	:
			00AB	31	MOVL	#1, @COMMENTLINE	:
08	00	6E	00	2C	BRW	14\$	:
			6E	00022	MOVCS	#0, (SP), #0, #8, TEMPDESC	: 1209
							:

57		6A	D0	00023	MOVL	TPARSE_BLOCK+8, STRINGSTART	1210
55	04	BC	3C	00026	MOVZWL	@INDESC, R5	1215
		55	D7	0002A	DECL	R5	
53	04	AC	D0	0002C	MOVL	INDESC, R3	1218
		52	D4	00030	CLRL	I	
		7D	11	00032	BRB	11\$	
50	04	B342	9A	00034	MOVZBL	@4(R3)[I], R0	1220
56		50	90	00039	MOVB	R0, CCHAR	
21		50	91	0003C	CMPB	R0, #33	
		06	12	0003F	BNEQ	4\$	
08	BC	01	D0	00041	MOVL	#1, @COMMENTLINE	1222
		72	11	00045	BRB	12\$	
2D		56	91	00047	CMPB	CCHAR, #45	1223
		5B	12	0004A	BNEQ	10\$	
55		52	D1	0004C	CMPL	I, R5	1226
		36	13	0004F	BEQL	9\$	
54		01	D0	00051	MOVL	#1, STATUS	1231
50		52	D0	00054	MOVL	I, J	1232
		24	11	00057	BRB	7\$	
51	04	B340	9A	00059	MOVZBL	@4(R3)[J], R1	1233
21		51	91	0005E	CMPB	R1, #33	
		09	12	00061	BNEQ	6\$	
54		01	D0	00063	MOVL	#1, STATUS	1235
08	BC	01	D0	00066	MOVL	#1, @COMMENTLINE	
		18	11	0006A	BRB	8\$	
20		51	91	0006C	CMPB	R1, #32	1237
		0C	13	0006F	BEQL	7\$	
09		51	91	00071	CMPB	R1, #9	1238
		07	13	00074	BEQL	7\$	
		54	D4	00076	CLRL	STATUS	1240
	08	BC	D4	00078	CLRL	@COMMENTLINE	
		07	11	0007B	BRB	8\$	
		50	D6	0007D	INCL	J	1233
55		50	D1	0007F	CMPL	J, R5	
		D5	1B	00082	BLEQU	5\$	
20		54	E9	00084	BLBC	STATUS, 10\$	1241
	00000000G	8F	DD	00087	PUSHL	#LINS PREMEOF	1249
		04	AE	9F	PUSHAB	TEMPDESC	
69		02	FB	00090	CALLS	#2, READOPTION	
58		50	D0	00093	MOVL	R0, RMSSTATUS	
	08	AC	DD	00096	PUSHL	COMMENTLINE	1250
	04	AE	9F	00099	PUSHAB	TEMPDESC	
01FC	C9	02	FB	0009C	CALLS	#2, SCANLINE	
	E2	08	BC	000A1	BLBS	@COMMENTLINE, 9\$	1252
		21	11	000A5	BRB	14\$	1254
7E		56	9A	000A7	MOVZBL	CCHAR, -(SP)	1263
01AE	C9	01	FB	000AA	CALLS	#1, STORECHARACTER	
		52	D6	000AF	INCL	I	1215
55		52	D1	000B1	CMPL	I, R5	
		03	1A	000B4	BGTRU	12\$	
		FF7B	31	000B6	BRW	3\$	
08	08	BC	E9	000B9	BLBC	@COMMENTLINE, 13\$	1265
57		6A	D1	000BD	CMPL	TPARSE_BLOCK+8, STRINGSTART	1266
		03	12	000C0	BNEQ	13\$	
		FF51	31	000C2	BRW	1\$	
	08	BC	D4	000C5	CLRL	@COMMENTLINE	1271
50		01	D0	000C8	MOVL	#1, R0	1272

LNK_PROCOPTIONS
V04=000

Linker options parser
Form complete options line, handling continuati

6 1
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 70
(5)

04 000CB

RET

; 1274

; Routine Size: 204 bytes, Routine Base: \$CODE\$ + 01FC


```
: 1166      1275 1 %sbttl 'TPARSE action routines';
: 1167      1276 1 routine set_base =
: 1168      1277 2   begin
: 1169      1278 2   ---
: 1170      1279 2   This routine is called by TPARSE to process the BASE= option
: 1171      1280 2
: 1172      1281 2   Inputs:
: 1173      1282 2
: 1174      1283 2       AP                      Points to tparse block
: 1175      1284 2       AP[tpa$l_number]      base as specified by user
: 1176      1285 2
: 1177      1286 2   Outputs:
: 1178      1287 2
: 1179      1288 2       Base stored in clu$l_base of the default cluster
: 1180      1289 2
: 1181      1290 2   ---
: 1182      1291 2   builtin
: 1183      1292 2   ap;
: 1184      1293 2   map
: 1185      1294 2   map ap : ref bblock;
: 1186      1295 2   if .lnk$gl_ctlmsk [lnk$v_sys]
: 1187      1296 2   then
: 1188      1297 2   begin
: 1189      1298 2       signal (lin$_optignsys);
: 1190      1299 2       return true
: 1191      1300 2   end;
: 1192      1301 2
: 1193      1302 2   lnk$gl_defclu [clu$l_base] = (.ap [tpa$l_number]      !Get value and round up
: 1194      1303 2   + 511) and not 511;                                     ! to a page boundary
: 1195      1304 2   lnk$gl_defclu [clu$v_based] = true;                  !Flag cluster is based
: 1196      1305 2   !lnk$gl_defclu [clu$v_usrbased] = true;              ! by user (not currently need for object clusters)
: 1197      1306 2   lnk$gl_ctlmsk [lnk$v_ubased] = true;                !Flag image based by user
: 1198      1307 2   return true
: 1199      1308 2   end;
: 1200      1309 1   end;                                           !Of set_base
```

```
0000 00000 SET_BASE:
OF 00000000G 00 00000000G 03 E1 00002 .WORD Save nothing : 1276
00000000G 00 00000000G 8F DD 0000A BBC #3, LNK$GL_CTLMSK, 1$ : 1296
00000000G 00 00000000G 01 FB 00010 PUSHL #LIN$ OPTIGNSYS : 1299
50 1C AC 000001FF 23 11 00017 CALLS #1, LIB$SIGNAL : 1300
00000000G 00 000001FF 8F C1 00019 1$: BRB 2$ : 1303
00000000G 00 000001FF 8F CB 00022 ADDL3 #511, 28(AP), R0 : 1304
00000000G 00 000001FF 01 88 0002E BICL3 #511, R0, LNK$GL_DEFCLU+76 : 1305
00000000G 00 000001FF 08 88 00035 BISB2 #1, LNK$GL_DEFCLU+88 : 1307
50 01 D0 0003C 2$: BISB2 #8, LNK$GL_CTLMSK+3 : 1308
04 0003F RET #1, R0 : 1309
```

; Routine Size: 64 bytes, Routine Base: \$CODE\$ + 02C8

```
1202 1310 1 routine set_channels =
1203 1311 2   begin
1204 1312 2   ---
1205 1313 2   This routine is called by TPARSE to process the CHANNELS= option
1206 1314 2
1207 1315 2   Inputs:
1208 1316 2
1209 1317 2       AP          Points to tparse_block
1210 1318 2       AP[tpa$l_number]  Number of channels as specified in options file
1211 1319 2
1212 1320 2   Outputs:
1213 1321 2
1214 1322 2       lnk$gw_chans      set up
1215 1323 2
1216 1324 2   ---
1217 1325 2   builtin
1218 1326 2   ap:
1219 1327 2   map
1220 1328 2   ap : ref bblock;
1221 1329 2   if (lnk$gw_chans = .ap [tpa$l_number]) gtru lnk$k_maxchans or .ap [tpa$l_number] eql 0
1222 1330 2   then
1223 1331 2       optionvaluerr (channels_name, .ap, 1, lnk$k_maxchans);
1224 1332 2   return true
1225 1333 2   end;
1226 1334 1
```

!Of set_channels

		0000 00000 SET_CHANNELS:				
				.WORD	Save nothing	: 1310
				MOVL	28(AP), R0	: 1330
00000000'	50	1C	AC D0 00002	MOVW	R0, LNK\$GW_CHANS	
00000FFF	EF		50 B0 00006	CMPL	R0, #4095	
			50 D1 0000D	BGTRU	1\$	
		1C	05 1A 00014	TSTL	28(AP)	
			AC D5 00016	BNEQ	2\$	
	7E	0FFF	16 12 00019	MOVZWL	#4095, -(SP)	: 1332
			8F 3C 0001B 1\$:	PUSHL	#1	
			01 DD 00020	PUSHL	AP	
			5C DD 00022	PUSHAB	CHANNELS_NAME	
00000000V	EF	00000000'	04 9F 00024	CALLS	#4, OPTIONVALUERR	
	50		01 FB 0002A	MOVL	#1, R0	: 1333
			01 D0 00031 2\$:	RET		: 1334
			04 00034			

; Routine Size: 53 bytes, Routine Base: \$CODE\$ + 0308


```
1228 1335 1 routine createcluster =
1229 1336 2   begin
1230 1337 2   ---
1231 1338 2
1232 1339 2   This routine is called by TPARSE to create the cluster descriptor
1233 1340 2   after the option has been parsed
1234 1341 2
1235 1342 2   Inputs:
1236 1343 2
1237 1344 2       AP                Points to tparse_block
1238 1345 2       AP[tpa$l_tokencnt] String descriptor for cluster name
1239 1346 2
1240 1347 2   Outputs:
1241 1348 2
1242 1349 2       New cluster descriptor allocated, cluster name set into descriptor.
1243 1350 2
1244 1351 2       lnk$gl_curclu    Pointer to new cluster descriptor
1245 1352 2       AP[tpa$l_number]  Zeroed
1246 1353 2
1247 1354 2   ---
1248 1355 2   builtin
1249 1356 2   ap;
1250 1357 2   map
1251 1358 2       ap : ref bblock;
1252 1359 2       namelengthcheck (ap [tpa$l_tokencnt], lnk$gt_clustering, lnk$sk_max_filename_length); !Check length of cl
1253 1360 2       lnk$alloccluster (lnk$gl_curclu); !Allocate cluster descriptor
1254 1361 2       cluoptflag = true; !Flag processing a cluster option
1255 1362 2
1256 1363 2   Move in the cluster name. Don't call LNK$INSERT_CLU here because the file specified
1257 1364 2   may contain the /SHARE qualifier. The cluster will be inserted in processfile
1258 1365 2
1259 1366 2       ch$move ((lnk$gl_curclu [clu$b_naming] = .ap [tpa$l_tokencnt]), .ap [tpa$l_tokenptr],
1260 1367 2       lnk$gl_curclu [clu$t_name]);
1261 1368 2       lnk$gl_curclu [clu$v_protect] = .protectflag; !Set protect flag as requested
1262 1369 2       ap [tpa$l_number] = 0; !Preset base to 0
1263 1370 2       return true
1264 1371 1   end; !Of createcluster
```

00FC 0000 CREATECLUSTER:

					57 00000000G	00 9E 00002	WORD	Save R2,R3,R4,R5,R6,R7	: 1335
					00000000G	8F DD 00009	MOVAB	LNK\$GL_CURCLU, R7	
					00000000G	00 9F 0000F	PUSHL	#LNK\$K_MAX_FILENAME_LENGTH	: 1359
					10	AC 9F 00015	PUSHAB	LNK\$GT_CLUSTRING	
						03 FB 00018	PUSHAB	16(AP)	
					00000000V	EF	CALLS	#3, NAMELENGTHCHECK	
						57 DD 0001F	PUSHL	R7	: 1360
					00000000G	00	CALLS	#1, LNK\$ALLOCLUSTER	
					00000000'	EF	CALLS	#1, LNK\$ALLOCLUSTER	
					56	01 FB 00021	MOVL	#1, CLUOPTFLAG	: 1361
					50	67 DO 0002F	MOVL	LNK\$GL_CURCLU, R6	: 1366
					5C	AC DO 00032	MOVL	16(AP), R0	
					A6	50 90 00036	MOVB	R0, 92(R6)	
					14	50 28 0003A	MOVC3	R0, @20(AP), 93(R6)	: 1367
58	A6	5D	A6	01	07 00000000'	EF FO 00040	INSV	PROTECTFLAG, #7, #1, 88(R6)	: 1368

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

K 1
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 74
(8)

50 1C AC D4 0004A CLRL 28(AP)
01 D0 0004D MOVL #1, R0
04 00050 RET

: 1369
: 1370
: 1371

; Routine Size: 81 bytes, Routine Base: \$CODE\$ + 033D

LN
VO


```
: 1266      1372 1 routine setclusterbase =
: 1267      1373 2   begin
: 1268      1374 2   ---
: 1269      1375 2   This routine is called by TPARSE when the base field of the cluster option is parsed
: 1270      1376 2
: 1271      1377 2   Inputs:
: 1272      1378 2
: 1273      1379 2
: 1274      1380 2       AP          pointer to tparse_block
: 1275      1381 2       AP[tpa$l_number] Base as specified in option
: 1276      1382 2       AP[tpa$l_tokencnt] Size of token just parsed (i.e. non-zero if base specified)
: 1277      1383 2
: 1278      1384 2   Outputs:
: 1279      1385 2
: 1280      1386 2       Cluster base and based flag set (iff base specified) for current cluster
: 1281      1387 2
: 1282      1388 2   ---
: 1283      1389 2   builtin
: 1284      1390 2   ap;
: 1285      1391 2   map
: 1286      1392 2       ap : ref bblock;
: 1287      1393 2       lnk$gl_curclu [clu$l_usrbase] = (.ap [tpa$l_number] + 511) !Set cluster base
: 1288      1394 2       and not 511; ! rounded up to page boundary
: 1289      1395 2       lnk$gl_curclu [clu$v_usrbased] = (.ap [tpa$l_tokencnt] neq 0);
: 1290      1396 2       ! If number present, then cluster is based
: 1291      1397 2       lnk$gl_ctlmsk [lnk$v_ubased] = .lnk$gl_ctlmsk [lnk$v_ubased] ! set if based by user
: 1292      1398 2       or .lnk$gl_curclu [clu$v_usrbased];
: 1293      1399 2       ap [tpa$l_number] = 0; !Zero in case pfc defaulted
: 1294      1400 2       return true
: 1295      1401 1   end; !Of setclusterbase
```

```
0004 0000 SETCLUSTERBASE:
: 1372      .WORD      Save R2
: 1393      MOVL      LNK$GL_CURCLU, R0
: 1394      ADDL3     #511, 28(AP), R1
: 1395      BICL3     #511, R1, 60(R0)
: 1396      CLRL     R1
: 1397      TSTL     16(AP)
: 1398      BEQL     1$, R1
: 1399      INCL     R1
: 1400      1$:      INSV     R1, #2, #1, 89(R0)
: 1401      EXTZV     #3, #1, LNK$GL_CTLMSK+3, R1
: 1402      EXTZV     #2, #1, 89(R0), R2
: 1403      BISB2     R2, R1
: 1404      INSV     R1, #3, #1, LNK$GL_CTLMSK+3
: 1405      CLRL     28(AP)
: 1406      MOVL     #1, R0
: 1407      RET
```

; Routine Size: 76 bytes, Routine Base: \$CODE\$ + 038E

```
: 1297      1402 1 routine setclusterpfc =
: 1298      1403 2   begin
: 1299      1404 2   ---
: 1300      1405 2   This routine is called by TPARSE to store the pfc in the cluster descriptor
: 1301      1406 2
: 1302      1407 2   Inputs:
: 1303      1408 2
: 1304      1409 2       AP          Pointer to tparse block
: 1305      1410 2       AP[tpa$l_number] Cluster page fault cluster factor
: 1306      1411 2
: 1307      1412 2   Outputs:
: 1308      1413 2
: 1309      1414 2       pfc field set in current cluster
: 1310      1415 2
: 1311      1416 2   ---
: 1312      1417 2   builtin
: 1313      1418 2       ap;
: 1314      1419 2   map
: 1315      1420 2       ap : ref bblock;
: 1316      1421 2       if (lnk$gl_curclu [clu$b_pfc] = .ap [tpa$l_number]) gtru lnk$k_maxpfc
: 1317      1422 2       then
: 1318      1423 2         optionvaluerr (clusterpfc_name, .ap, 0, lnk$k_maxpfc);
: 1319      1424 2       return true
: 1320      1425 2   end;
: 1321      1426 1
```

```
0000 00000 SETCLUSTERPFC:
      51 00000000G 00 D0 00002 .WORD Save nothing : 1402
      50          1C AC D0 00009 MOVL LNK$GL_CURCLU, R1 : 1422
      5A A1 8F 50 90 0000D MOVL 28(AP), R0
000000FF 8F 50 D1 00011 MOVBL R0, 90(R1)
      7E FF 15 1B 00018 CMPL R0, #255
      7E FF 8F 9A 0001A BLEQU 1$
      7E FF 7E D4 0001E MOVZBL #255, -(SP) : 1424
      7E FF 5C DD 00020 CLRL -(SP)
      00000000V EF 00000000' EF 9F 00022 PUSHL AP
      50          04 FB 00028 PUSHAB CLUSTERPFC NAME
      01 D0 0002F 1$ CALLS #4, OPTIONVALUERR
      04 00032 MOVL #1, R0 : 1425
      RET : 1426
```

; Routine Size: 51 bytes, Routine Base: \$CODE\$ + 03DA


```

: 1323      1427 1 routine clusterdone =
: 1324      1428 2   begin
: 1325      1429 2   |
: 1326      1430 2   | This routine finishes cluster option processing
: 1327      1431 2   |
: 1328      1432 2   | if not .lnk$gl_curclu [clu$v_shring]      !If not a shareable image cluster
: 1329      1433 2   |   and .lnk$gl_curclu [clu$v_usrbased]    ! and it was based by user
: 1330      1434 2   | then
: 1331      1435 2   |   begin
: 1332      1436 2   |     lnk$gl_curclu [clu$l_base] = .lnk$gl_curclu [clu$l_usrbase];    !Set cluster base
: 1333      1437 2   |     lnk$gl_curclu [clu$l_usrbase] = 0;      !Clear usrbase
: 1334      1438 2   |     lnk$gl_curclu [clu$v_based] = true;      !Flag cluster as based
: 1335      1439 2   |   end;
: 1336      1440 2   | return true
: 1337      1441 1   end;

```

				0000 00000 CLUSTERDONE:				
		50	000000000G	00	D0 00002	.WORD	Save nothing	: 1427
11	58	A0		02	E0 00009	MOVL	LNK\$GL_CURCLU, R0	: 1432
0C	59	A0		02	E1 0000E	BBS	#2, 88(R0), 1\$	
	4C	A0	3C	A0	D0 00013	BBC	#2, 89(R0), 1\$	: 1433
			3C	A0	D4 00018	MOVL	60(R0), 76(R0)	: 1436
	58	A0		01	88 0001B	CLRL	60(R0)	: 1437
	50			01	D0 0001F 1\$:	BISB2	#1, 88(R0)	: 1438
				04	00022	MOVL	#1, R0	: 1440
						RET		: 1441

; Routine Size: 35 bytes, Routine Base: \$CODE\$ + 040D

```
: 1339      1442 1 routine set_collect =
: 1340      1443 2   begin
: 1341      1444 2   ---
: 1342      1445 2
: 1343      1446 2   This routine is called by TPARSE when the cluster name field of the COLLECT option is parsed
: 1344      1447 2
: 1345      1448 2   Inputs:
: 1346      1449 2
: 1347      1450 2       AP          Pointer to tparse_block
: 1348      1451 2       AP[tpa$l_tokencnt] string descriptor for cluster name
: 1349      1452 2
: 1350      1453 2   Outputs:
: 1351      1454 2
: 1352      1455 2       curcollectdesc Pointer to allocated collect descriptor
: 1353      1456 2
: 1354      1457 2   ---
: 1355      1458 2   builtin
: 1356      1459 2   ap;
: 1357      1460 2   map
: 1358      1461 2       ap : ref bblock;
: 1359      1462 2       namelengthcheck (ap [tpa$l_tokencnt], lnk$gt_clustering, lnk$k_max_filename_length); !Check cluster name
: 1360      1463 2       search_insert_list (lnk$gl_cclulst, ap [tpa$l_tokencnt],
: 1361      1464 2                               !Insert / lookup the cluster in the cluster collect list
: 1362      1465 2               ccd$c_size, curcollectdesc);
: 1363      1466 2       curcollectdesc [ccd$b_protect] = .protectflag;      !Propagate the cluster protected flag
: 1364      1467 2       return true
: 1365      1468 1       end;
                                !Of set_collect
```

```
                                0004 00000 SET_COLLECT:
                                .WORD      Save R2
                                MOVAB      CURCOLLECTDESC, R2
                                PUSHAB     #LNK$K_MAX_FILENAME_LENGTH
                                PUSHAB     LNK$GT_CLUSTERING
                                PUSHAB     16(AP)
                                CALLS      #3, NAMELENGTHCHECK
                                PUSHAB     16(AP)
                                PUSHAB     LNK$GL_CCLULST
                                CALLS      #4, SEARCH_INSERT_LIST
                                MOVL       CURCOLLECTDESC, R0
                                MOVB      PROTECTFLAG, 48(R0)
                                MOVL      #1, R0
                                RET
                                : 1442
                                : 1462
                                : 1463
                                : 1466
                                : 1467
                                : 1468
```

; Routine Size: 63 bytes, Routine Base: \$CODE\$ + 0430


```
: 1367      1469 1 routine collect_psect =
: 1368      1470 2   begin
: 1369      1471 2   ---
: 1370      1472 2
: 1371      1473 2   This routine is called by TPARSE for each psect in the COLLECT option
: 1372      1474 2
: 1373      1475 2   Inputs:
: 1374      1476 2
: 1375      1477 2       AP          Pointer to tparse_block
: 1376      1478 2       AP[tpa$l_tokencnt] String descriptor for psect name
: 1377      1479 2
: 1378      1480 2   Outputs:
: 1379      1481 2
: 1380      1482 2       Psect name block hung off of currcollectdesc in name order.
: 1381      1483 2
: 1382      1484 2   ---
: 1383      1485 2   builtin
: 1384      1486 2   ap;
: 1385      1487 2   map
: 1386      1488 2   ap : ref bblock;
: 1387      1489 2   local
: 1388      1490 2   scratchptr;
: 1389      1491 2   namelengthcheck (ap [tpa$l_tokencnt], lnk$gt_pscstring); !Check psect name length
: 1390      1492 2   search_insert_list (curcollectdesc [ccd$l_psc1st], ap [tpa$l_tokencnt],
: 1391      1493 2   !Lookup/insert psect into list for cluster
: 1392      1494 2   cpd$size, scratchptr);
: 1393      1495 2   return true
: 1394      1496 1   end;                                     !Of collect_psect
```

```
0000 00000 COLLECT_PSECT:
      SE      00000000G 04 C2 00002 .WORD Save nothing      : 1469
      10      AC 9F 00005 00 9F 00005 SUBL2 #4, SP          :
      00000000V EF 02 FB 0000B AC 9F 0000B PUSHAB LNK$GT_PSCSTRING : 1491
      10      5E DD 00015 02 FB 0000B PUSHAB 16(AP)
      7E 00000000' EF 24 DD 00017 5E DD 00015 CALLS #2, NAMELENGTHCHECK :
      00000000V EF 2C C1 0001C 24 DD 00017 PUSHL SP              : 1492
      50      04 FB 00024 AC 9F 00019 PUSHL #36
      01      04 0002E 2C C1 0001C PUSHAB 16(AP)
      04 0002E 04 FB 00024 ADDL3 #44, CURCOLLECTDESC, -(SP)
      04 0002E 01 D0 0002B CALLS #4, SEARCH_INSERT_LIST
      04 0002E 04 0002E MOVL #1, R0
      RET                                : 1495
      : 1496
```

; Routine Size: 47 bytes, Routine Base: \$CODE\$ + 046F

```
: 1396      1497 1 routine set_min_dzro =
: 1397      1498 2   begin
: 1398      1499 2   |
: 1399      1500 2   | This routine is called by TPARSE to set the DZRO_MIN= option
: 1400      1501 2   |
: 1401      1502 2   |   builtin
: 1402      1503 2   |   ap;
: 1403      1504 2   |   map
: 1404      1505 2   |   ap : ref bblock;
: 1405      1506 2   |
: 1406      1507 2   |   if (lnk$gw_dzromin = .ap [tpa$l_number]) gtru lnk$k_mindzro
: 1407      1508 2   |   then
: 1408      1509 2   |       optionvaluerr (dzromin_name, .ap, 0,
: 1409      1510 2   |           lnk$k_mindzro);
: 1410      1511 2   |   if .lnk$gl_ctlmsk [lnk$v_sys] then signal (lin$optignsys);
: 1411      1512 2   |   return true
: 1412      1513 1   end;
                                !Of set_min_dzro
```

				0000 00000 SET_MIN_DZRO:				
						.WORD	Save nothing	: 1497
						MOVW	28(AP), R0	: 1507
00000000G	00	1C	AC	D0	00002	MOVW	R0, LNK\$GW_DZROMIN	
0000FFFF	8F		50	B0	00006	CMPL	R0, #65535	
			16	1B	00014	BLEQU	1\$	
	7E	FFFF	8F	3C	00016	MOVZWL	#65535, -(SP)	: 1509
			7E	D4	0001B	CLRL	-(SP)	
			5C	DD	0001D	PUSHL	AP	
		00000000'	EF	9F	0001F	PUSHAB	DZROMIN NAME	
00000000V	EF		04	FB	00025	CALLS	#4, OPTIONVALUERR	
0D 00000000G	00		03	E1	0002C	BBC	#3, LNK\$GL_CTLMSK, 2\$	: 1511
		00000000G	8F	DD	00034	PUSHL	#LIN\$ OPTIGNSYS	
00000000G	00		01	FB	0003A	CALLS	#1, LIB\$SIGNAL	
	50		01	D0	00041	MOVW	#1, R0	: 1512
			04	00	00044	RET		: 1513

; Routine Size: 69 bytes, Routine Base: \$CODE\$ + 049E


```

: 1414      1514 1 routine set_gsmatch_ctl =
: 1415      1515 2   begin
: 1416      1516 2   |
: 1417      1517 2   | This routine is called by TPARSE when the match control field of the GSMATCH option is processed
: 1418      1518 2   |
: 1419      1519 2   builtin
: 1420      1520 2   ap;
: 1421      1521 2   map
: 1422      1522 2   ap : ref bblock;
: 1423      1523 2   lnk$gb_matchctl = .ap [tpa$l_param];      !Match control is passed as a parameter
: 1424      1524 2   ap [tpa$l_number] = 0;                  !Preset GSMATCH
: 1425      1525 2   return true
: 1426      1526 1   end;                                     !Of set_gsmatch_ctl

```

```

                                0000 00000 SET_GSMATCH_CTL:
                                .WORD  Save nothing
00000000G 00          20 AC 90 00002      MOVB 32(AP), LNK$GB_MATCHCTL      : 1514
                                1C AC D4 0000A      CLRL 28(AP)                  : 1523
                                50          01 D0 0000D      MOVL #1, R0          : 1524
                                04 00010      RET                                : 1525
                                                : 1526

```

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 04E3

```
: 1428      1527 1 routine set_gsmatch_maj =
: 1429      1528 2   begin
: 1430      1529 2   |
: 1431      1530 2   | This routine is called by TPARSE to process the major id of the GSMATCH option
: 1432      1531 2   |
: 1433      1532 2   |   builtin
: 1434      1533 2   |   ap;
: 1435      1534 2   |   map
: 1436      1535 2   |   ap : ref bblock;
: 1437      1536 2   |
: 1438      1537 2   |   if (lnk$gl_matchid [gmt$b_majorid] = .ap [tpa$l_number]) gtru lnk$k_maxmajid
: 1439      1538 2   |   then
: 1440      1539 2   |       optionvaluerr (gsmatch_name, .ap, 0, lnk$k_maxmajid);
: 1441      1540 2   |   ap [tpa$l_number] = 0;
: 1442      1541 2   |   return true
: 1443      1542 1   | end;
                                !Of set_gsmatch_maj
```

0000 00000 SET_GSMATCH MAJ:									
						.WORD	Save nothing		1527
						MOVL	28(AP), R0		1537
00000000G	00	1C	AC	D0	00002	MOVB	R0, LNK\$GL_MATCHID+3		
000000FF	8F		50	D1	0000D	CMP	R0, #255		
			15	1B	00014	BLEQU	1\$		
	7E	FF	8F	9A	00016	MOVZBL	#255, -(SP)		1539
			7E	D4	0001A	CLRL	-(SP)		
			5C	DD	0001C	PUSHL	AP		
		00000000'	EF	9F	0001E	PUSHAB	GSMATCH NAME		
00000000V	EF		04	FB	00024	CALLS	#4, OPTIONVALUERR		
		1C	AC	D4	0002B	CLRL	28(AP)		1540
	50		01	D0	0002E	MOVL	#1, R0		1541
			04	00031		RET			1542

; Routine Size: 50 bytes, Routine Base: \$CODE\$ + 04F4


```
: 1445      1543 1 routine set_gsmatch_min =
: 1446      1544 2   begin
: 1447      1545 2   |
: 1448      1546 2   | This routine is called by TPARSE to process the minor id of the GSMATCH option
: 1449      1547 2   |
: 1450      1548 2   |   builtin
: 1451      1549 2   |   ap;
: 1452      1550 2   |   map
: 1453      1551 2   |   ap : ref bblock;
: 1454      1552 2   |   if (lnk$gl_matchid [gmt$b_minorid] = .ap [tpa$l_number]) gtru lnk$k_maxminid
: 1455      1553 2   |   then
: 1456      1554 2   |       optionvaluerr (gsmatch_name, .ap, 0, lnk$k_maxminid);
: 1457      1555 2   |   return true;
: 1458      1556 1   end;
                                !Of set_gsmatch_min
```

```
00000000G 00      18 00FFFFFF 8F      00FFFFFF 00000000' 00000000V EF 50      01 D0 0002F 1$:
                                0000 00000 SET_GSMATCH_MIN:
                                .WORD Save nothing
                                MOVL 28(AP), R0
                                INSV R0, #0, #24, LNK$GL_MATCHID
                                CMPL R0, #16777215
                                BLEQU 1$
                                PUSHL #16777215
                                CLRL -(SP)
                                PUSHL AP
                                PUSHAB GSMATCH_NAME
                                CALLS #4, OPTIONVALUERR
                                MOVL #1, R0
                                RET
```

; Routine Size: 51 bytes, Routine Base: \$CODE\$ + 0526

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

H 2
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 84
(18)

```
: 1460      1557 1 routine set_image_type =  
: 1461      1558 2   begin  
: 1462      1559 2   |  
: 1463      1560 2   | This routine sets the image type.  
: 1464      1561 2   |  
: 1465      1562 2   |   lnk$gl_ctlmsk [lnk$y_cli] = true;  
: 1466      1563 2   |   return true  
: 1467      1564 1   end;
```

```
0000 00000 SET_IMAGE TYPE:  
00000000G 00      20 88 00002   .WORD   Save nothing  
50          01 00 00009   BISB2   #32, LNK$GL_CTLMSK+3  
          04 0000C   MOVL    #1, R0  
          RET
```

```
: 1557  
: 1562  
: 1563  
: 1564
```

; Routine Size: 13 bytes, Routine Base: \$CODE\$ + 0559


```
: 1469 1565 1 routine set_image_ident =
: 1470 1566 2   begin
: 1471 1567 2   :
: 1472 1568 2   : This routine is called by TPARSE to set the image file identification
: 1473 1569 2   :
: 1474 1570 2   builtin
: 1475 1571 2   ap;
: 1476 1572 2   map
: 1477 1573 2   ap : ref bblock;
: 1478 1574 2   local
: 1479 1575 2   ptr;
: 1480 1576 2   delimited_ident;
: 1481 1577 2   ident_desc : vector [2];
: 1482 1578 2
: 1483 1579 2   lnk$gl_ctlmsk [lnk$v_imgidopt] = true;      ! Flag that image id is manually set
: 1484 1580 2   if (delimited_ident = .ap [tpa$l_param])
: 1485 1581 2   then                                     ! Yes, then better find matching end delimiter
: 1486 1582 2       if (ptr = ch$find_ch (.ap [tpa$l_stringcnt], .ap [tpa$l_stringptr], .ap [tpa$l_char])) eql 0
: 1487 1583 2       then
: 1488 1584 2           return false                      ! If no end delimiter match, balk
: 1489 1585 2       else
: 1490 1586 2           begin                               ! If found one, adjust pointers accordingly
: 1491 1587 2               ident_desc [0] = .ptr - .ap [tpa$l_stringptr];
: 1492 1588 2               ident_desc [1] = .ap [tpa$l_stringptr];
: 1493 1589 2               ap [tpa$l_stringcnt] = .ap [tpa$l_stringcnt] - (.ident_desc [0] + 1);
: 1494 1590 2               ap [tpa$l_stringptr] = .ptr + 1;
: 1495 1591 2           end
: 1496 1592 2       else
: 1497 1593 2           begin                               ! If not delimited,
: 1498 1594 2               ident_desc [0] = .ap [tpa$l_tokencnt];      ! then token points to
: 1499 1595 2               ident_desc [1] = .ap [tpa$l_tokenptr];    ! the ident string
: 1500 1596 2           end;
: 1501 1597 2
: 1502 1598 2   namelengthcheck (ident_desc [0], image_ident_name, 15); ! make sure legal length
: 1503 1599 2   lnk$gt_imgid [0] = .ident_desc [0];      ! fill in image id field
: 1504 1600 2   ch$move (.ident_desc [0], .ident_desc [1], lnk$gt_imgid [1]);
: 1505 1601 2   return true
: 1506 1602 1   end;
```

003C 00000 SET_IMAGE_IDENT:

			5E		08	C2	00002	WORD	Save R2,R3,R4,R5	: 1565
		00000000G	00	40	8F	88	00005	SUBL2	#8, SP	: 1579
			50	20	AC	D0	0000D	BISB2	#64, LNK\$GL_CTLMSK+3	: 1580
			2A		50	E9	00011	MOVL	32(AP), DELIMITED_IDENT	: 1582
0C	BC	08	AC	18	AC	3A	00014	BLBC	DELIMITED_IDENT, 2\$	
					02	12	0001B	LOCC	24(AP), 8(AP), @12(AP)	
					51	D4	0001D	BNEQ	1\$	
					51	D5	0001F	CLRL	R1	
					51	D5	0001F	TSTL	PTR	
					45	13	00021	BEQL	4\$	
	6E		51	0C	AC	C3	00023	SUBL3	12(AP), PTR, IDENT_DESC	: 1587
		04	AE	0C	AC	D0	00028	MOVL	12(AP), IDENT_DESC+4	: 1588
	50	08	AC		6E	C3	0002D	SUBL3	IDENT_DESC, 8(AP), R0	: 1589

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

J 2
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 86
(19)

08	AC	FF	A0	9E	00032	MOVAB	-1(R0), 8(AP)	:	
0C	AC	01	A1	9E	00037	MOVAB	1(R1), 12(AP)	:	1590
			04	11	0003C	BRB	3\$	:	1582
	6E	10	AC	7D	0003E	MOVQ	16(AP), IDENT_DESC	:	1594
			0F	DD	00042	PUSHL	#15	:	1598
		00000000'	EF	9F	00044	PUSHAB	IMAGE_IDENT_NAME	:	
		08	AE	9F	0004A	PUSHAB	IDENT_DESC	:	
	00000000V	EF	03	FB	0004D	CALLS	#3, NAMELENGTHCHECK	:	
	00000000G	00	6E	90	00054	MOVB	IDENT_DESC, LNK\$GT_IMGID	:	1599
00000000G	00	04	6E	28	0005B	MOVC3	IDENT_DESC, @IDENT_DESC+4, LNK\$GT_IMGID+1	:	1600
			01	D0	00064	MOVL	#1, R0	:	1601
				04	00067	RET		:	
			50	D4	00068	CLRL	R0	:	1602
				04	0006A	RET		:	

; Routine Size: 107 bytes, Routine Base: \$CODE\$ + 0566

LN
V0


```
: 1508      1603 1 routine set_image_name =  
: 1509      1604 2   begin  
: 1510      1605 2  
: 1511      1606 2   This routine is called by TPARSE to set the image name  
: 1512      1607 2  
: 1513      1608 2   builtin  
: 1514      1609 2   ap;  
: 1515      1610 2   map  
: 1516      1611 2   ap : ref bblock;  
: 1517      1612 2  
: 1518      1613 2   namelengthcheck (ap [tpa$l_tokencnt], image_name_name, lnk$k_max_filename_length); ! make sure legal le  
: 1519      1614 2   lnk$gt_imgnam [0] = .ap [tpa$l_tokencnt]; ! fill in image name field  
: 1520      1615 2   ch$move (.lnk$gt_imgnam [0], .ap [tpa$l_tokenptr], lnk$gt_imgnam [1]);  
: 1521      1616 2   return true  
: 1522      1617 1   end;
```

```
                                007C 00000 SET_IMAGE_NAME:  
                                .WORD Save R2,R3,R4,R5,R6  
                                MOVAB LNK$GT_IMGNAME, R6  
                                PUSHB #LNK$K_MAX_FILENAME_LENGTH  
                                PUSHAB IMAGE_NAME_NAME  
                                PUSHAB 16(AP)  
                                CALLS #3, NAMELENGTHCHECK  
                                MOVAB 16(AP), LNK$GT_IMGNAME  
                                MOVZBL LNK$GT_IMGNAME, R0  
                                MOVCL R0, @20(AP), LNK$GT_IMGNAME+1  
                                MOVL #1, R0  
                                RET  
                                : 1603  
                                : 1613  
                                : 1614  
                                : 1615  
                                : 1616  
                                : 1617
```

; Routine Size: 48 bytes, Routine Base: \$CODE\$ + 05D1

```
: 1524      1618 1 routine set_ioseg =
: 1525      1619 2   begin
: 1526      1620 2   |
: 1527      1621 2   | This routine is called by TPARSE to set the IOSEG= option
: 1528      1622 2   |
: 1529      1623 2   |   builtin
: 1530      1624 2   |       ap;
: 1531      1625 2   |   map
: 1532      1626 2   |       ap : ref bblock;
: 1533      1627 2   |   if (lnk$gw_ioseg = .ap [tpa$l_number]) gtru lnk$k_maxioseg
: 1534      1628 2   |   then
: 1535      1629 2   |       optionvaluerr (ioseg_name, .ap, 0,
: 1536      1630 2   |           lnk$k_maxioseg);
: 1537      1631 2   |   return true
: 1538      1632 1   end;                                !Of set_ioseg
```

```
0000 00000 SET_IOSEG:
00000000' 50      1C  AC  D0 00002  .WORD  Save nothing      : 1618
0000FFFF  EF      50  B0 00006  MOVL  28(AP), R0      : 1627
0000FFFF  8F      50  D1 0000D  MOVW  R0, LNK$GW_IOSEG
0000FFFF  7E      FF  16  1B 00014  CMPL  R0, #65535-
0000FFFF  7E      FF  8F  3C 00016  BLEQU 1$
0000FFFF  7E      FF  7E  D4 0001B  MOVZWL #65535, -(SP)      : 1629
0000FFFF  7E      FF  5C  DD 0001D  CLRL  -(SP)
0000FFFF  7E      FF  5C  DD 0001D  PUSHL  AP
0000FFFF  7E      FF  EF  9F 0001F  PUSHAB IOSEG_NAME
0000FFFF  7E      FF  04  FB 00025  CALLS #4, OPTIONVALUERR
0000FFFF  7E      FF  01  D0 0002C 1$: MOVL  #1, R0      : 1631
0000FFFF  7E      FF  01  D0 0002C 1$: RET      : 1632
```

; Routine Size: 48 bytes, Routine Base: \$CODE\$ + 0601

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

M 2
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 89
(22)

```
: 1540      1633 1 routine set_p0bufs =
: 1541      1634 2   begin
: 1542      1635 2   |
: 1543      1636 2   | Clear the NOP0BUFS flag
: 1544      1637 2   |
: 1545      1638 2   builtin
: 1546      1639 2   ap;
: 1547      1640 2   map
: 1548      1641 2   ap : ref bblock;
: 1549      1642 2
: 1550      1643 2   lnk$gl_ctlmsk [lnk$v_nop0bufs] = .ap [tpa$l_param];
: 1551      1644 2   return true
: 1552      1645 1   end;                                !0f p0bufs_on
```

										0000 00000 SET_POBUFS:		
00000000G	00		01		05	20	AC	F0	00002	.WORD	Save nothing	: 1633
				50			01	D0	0000C	INSV	32(AP), #5, #1, LNK\$GL_CTLMSK+2	: 1643
								04	0000F	MOVL	#1, R0	: 1644
										RET		: 1645

; Routine Size: 16 bytes, Routine Base: \$CODE\$ + 0631

```
: 1554      1646 1 routine set_isd_max =
: 1555      1647 2   begin
: 1556      1648 2   |
: 1557      1649 2   | This routine is called by TPARSE to set the ISD_MAX= option
: 1558      1650 2   |
: 1559      1651 2   |   builtin
: 1560      1652 2   |       ap;
: 1561      1653 2   |   map
: 1562      1654 2   |       ap : ref bblock;
: 1563      1655 2   |
: 1564      1656 2   |   if (lnk$gw_misects = .ap [tpa$l_number]) gtru lnk$k_maxisds or .lnk$gw_misects eql 0
: 1565      1657 2   |   then
: 1566      1658 2   |       optionvaluerr (isdmax_name, .ap, 1, lnk$k_maxisds);
: 1567      1659 2   |   if .lnk$gl_ctlmsk [lnk$v_shr] then signal (lin$optignshr);
: 1568      1660 2   |   if .lnk$gl_ctlmsk [lnk$v_sys] then signal (lin$optignsys);
: 1569      1661 2   |   return true
: 1570      1662 1   end;                                !Of set_isd_max
```

```
001C 00000 SET_ISD_MAX:
54 00000000G 00 9E 00002 .WORD Save R2,R3,R4 : 1646
53 00000000G 00 9E 00009 MOVAB LNK$GW_MISECTS, R4
52 00000000G 00 9E 00010 MOVAB LIB$SIGNAL, R3
50 1C AC D0 00017 MOVAB LNK$GL_CTLMSK, R2
64 8F 50 B0 0001B MOVAB 28(AP), R0 : 1656
0000FFFF 8F 50 D1 0001E CMPL R0, #65535
04 1A 00025 BGTRU 1$
64 B5 00027 TSTW LNK$GW_MISECTS
16 12 00029 BNEQ 2$
7E FFFF 8F 3C 0002B 1$: MOVZWL #65535, -(SP) : 1658
01 DD 00030 PUSHL #1
5C DD 00032 PUSHL AP
00000000' EF 9F 00034 PUSHAB ISD_MAX_NAME
09 62 04 FB 0003A CALLS #4, OPTIONVALUERR
00000000G 02 E1 00041 2$: BBC #2, LNK$GL_CTLMSK, 3$ : 1659
63 8F DD 00045 PUSHL #LIN$ OPTIGNSHR
09 62 01 FB 0004B CALLS #1, LIB$SIGNAL : 1660
00000000G 03 E1 0004E 3$: BBC #3, LNK$GL_CTLMSK, 4$
63 8F DD 00052 PUSHL #LIN$ OPTIGNSYS
50 01 FB 00058 CALLS #1, LIB$SIGNAL : 1661
01 D0 0005B 4$: MOVL #1, R0 : 1662
04 0005E RET
```

; Routine Size: 95 bytes, Routine Base: \$CODE\$ + 0641


```

: 1572      1663 1 routine set_protect =
: 1573      1664 2   begin
: 1574      1665 2   |
: 1575      1666 2   | This action routine sets the protect flag
: 1576      1667 2   |
: 1577      1668 2   |   builtin
: 1578      1669 2   |   ap;
: 1579      1670 2   |   map
: 1580      1671 2   |   ap : ref bblock;
: 1581      1672 2   |   protectflag = .ap [tpa$l_param];
: 1582      1673 2   |   return true
: 1583      1674 1   |   end;
                                !Of set_protect

```

```

                                0000 00000 SET_PROTECT:
                                .WORD
00000000' EF      20 AC D0 00002      MOVL      Save nothing      : 1663
                                01 D0 0000A      MOVL      32(AP), PROTECTFLAG : 1672
                                04 0000D      RET      #1, R0                  : 1673
                                                                : 1674

```

; Routine Size: 14 bytes, Routine Base: \$CODE\$ + 06A0

003C 00000 CREATEPSECT:

		SE	00000000G	24 C2 00002	.WORD	Save R2,R3,R4,R5	: 1675
			10 AC 9F 0C005	SUBL2	#36, SP	:	1688
			AC 9F 0000B	PUSHAB	LNK\$GT_PSCSTRING	:	
			02 FB 0000E	PUSHAB	16(AP)	:	
		EF	10 AC 90 00015	CALLS	#2, NAMELENGTHCHECK	:	
05 AE	04	AE	10 AC 28 0001A	MOVB	16(AP), PSECTNAMEBUF	: 1689	
	14	BC	EF 9F 00021	MOVCB	16(AP), @20(AP), PSECTNAMEBUF+1	: 1690	
			00000000'	PUSHAB	CURPSECTDESC	: 1691	
			00000000G	PUSHAB	LNK\$ALLOC_PDD	:	
			00000000G	PUSHAB	LNK\$COMPARE_PDD	:	
			OC AE D4 00033	CLRL	12(SP)	:	
			OC AE 9F 00036	PUSHAB	12(SP)	:	
			14 AE 9F 00039	PUSHAB	PSECTNAMEBUF	:	
			00000000'	PUSHAB	LNK\$GL_PSCDFLST	:	
			EF 9F 0003C	PUSHAB	LNK\$GL_PSCDFLST	:	
			06 FB 00042	CALLS	#6, LIB\$INSERT_TREE	:	
			01 D0 00049	MOVL	#1, R0	: 1692	
			04 0004C	RET		: 1693	

; Routine Size: 77 bytes, Routine Base: \$CODES + 06AE


```
1605 1694 1 routine set_pscatrib =
1606 1695 2   begin
1607 1696 2   |
1608 1697 2   | This routine is called by TPARSE for each attribute found in the PSECT= option
1609 1698 2   |
1610 1699 2   |   builtin
1611 1700 2   |   ap;
1612 1701 2   |   map
1613 1702 2   |   ap : ref bblock;
1614 1703 2   |   local
1615 1704 2   |   maskflag,
1616 1705 2   |   attribit;
1617 1706 2   |   attribit = .ap [tpa$l_param];
1618 1707 2   |
1619 1708 2   | Check if this is psect alignment and process
1620 1709 2   |
1621 1710 2   |   if (.attribit and sign_bit) neq 0
1622 1711 2   |   then
1623 1712 2   |   begin
1624 1713 2   |   curpsectdesc [pdd$b_align] = .attribit<0, 8>;   !Set psect alignment for later
1625 1714 2   |   return true;
1626 1715 2   |   end
1627 1716 2   |   else
1628 1717 2   |   if (.attribit and word_sign_bit) neq 0   !If clearing the bit
1629 1718 2   |   then
1630 1719 2   |   begin
1631 1720 2   |   curpsectdesc [pdd$w_flags] = .curpsectdesc [pdd$w_flags] and not ( not (.attribit and %x'FFFF'))
1632 1721 2   |   curpsectdesc [pdd$w_flgmsk] = .curpsectdesc [pdd$w_flgmsk] or ( not (.attribit and %x'FFFF'));
1633 1722 2   |   end
1634 1723 2   |   else
1635 1724 2   |   begin
1636 1725 2   |   curpsectdesc [pdd$w_flags] = .curpsectdesc [pdd$w_flags] or .attribit;
1637 1726 2   |   curpsectdesc [pdd$w_flgmsk] = .curpsectdesc [pdd$w_flgmsk] or .attribit;
1638 1727 2   |   end;
1639 1728 2   |   return true
1640 1729 1   end;

                                !Of set_pscatrib
```

```
                                0004 00000 SET_PSCATRIB:
                                .WORD   Save R2
                                50      20   AC   D0 00002   MOVL   32(AP), ATTRIBIT      : 1694
                                51 00000000' EF   D0 00006   MOVL   CURPSECTDESC, R1      : 1706
                                50      06   D5 0000D   TSTL   ATTRIBIT                  : 1713
                                06      18   0000F   BGEQ    1$                          : 1710
                                50      90   00011   MOVB    ATTRIBIT, 14(R1)           : 1713
                                23      11   00015   BRB     3$                          : 1714
                                50      B5   00017 1$:   TSTW   ATTRIBIT                  : 1717
                                17      18   00019   BGEQ    2$                          :
                                50      B2   0001B   MCOMW   ATTRIBIT, R2                : 1720
                                OA      A1   52      AA 0001E   BICW2  R2, 10(R1)         :
                                50      3C   00022   MOVZWL  ATTRIBIT, R0                : 1721
                                52      0C   A1      3C 00025   MOVZWL  12(R1), R2
                                50      52   CA 00029   BICL2  R2, R0
                                OC      A1   50      B2 0002C   MCOMW   R0, 12(R1)
```

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

E 3
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 94
(26)

0A A1
0C A1
50

08 11 00030
50 A8 00032 2\$:
50 A8 00036
01 D0 0003A 3\$:
04 0003D

BRB 3\$
BISW2 ATTRIBIT, 10(R1)
BISW2 ATTRIBIT, 12(R1)
MOVL #1, R0
RET

: 1717
: 1725
: 1726
: 1728
: 1729

; Routine Size: 62 bytes, Routine Base: \$CODE\$ + 06FB


```

: 1642      1730 1 routine set_shl =
: 1643      1731 2   begin
: 1644      1732 2   |
: 1645      1733 2   | This routine is called by TPARSE for the SHL_EXTRA option
: 1646      1734 2   |
: 1647      1735 2   builtin
: 1648      1736 2   ap;
: 1649      1737 2   map
: 1650      1738 2   ap : ref bblock;
: 1651      1739 2   lnk$gl_shlextra = .ap [tpa$l_number];
: 1652      1740 2   if .lnk$gl_ctlmsk [lnk$v_sys] then signal (lin$_optignsys);
: 1653      1741 2   return true
: 1654      1742 1   end;

```

				0000 00000	SET_SHL:.WORD	Save nothing	: 1730
	00000000'	EF	1C	AC D0 00002	MOVL	28(AP), LNK\$GL SHLEXTRA	: 1739
OD	00000000G	00		03 E1 0000A	BBC	#3, LNK\$GL_CTLMSK, 1\$	: 1740
			00000000G	8F DD 00012	PUSHL	#LIN\$ OPTIGNSYS	:
	00000000G	00		01 FB 00018	CALLS	#1, LIB\$SIGNAL	:
		50		01 D0 0001F	MOVL	#1, R0	: 1741
				04 00022	RET		: 1742

; Routine Size: 35 bytes, Routine Base: \$CODE\$ + 0739

```
: 1656      1743 1 routine set_stack =  
: 1657      1744 2   begin  
: 1658      1745 2   :  
: 1659      1746 2   : This routine is called by TPARSE to process the STACK= option  
: 1660      1747 2   :  
: 1661      1748 2   builtin  
: 1662      1749 2   ap;  
: 1663      1750 2   map  
: 1664      1751 2   ap : ref bblock;  
: 1665      1752 2   if (lnk$gw_stack = .ap [tpa$l_number]) gtru lnk$k_maxstack or .lnk$gw_stack eql 0  
: 1666      1753 2   then  
: 1667      1754 2   optionvaluerr (stack_name, .ap, 1, lnk$k_maxstack);  
: 1668      1755 2   if .lnk$gl_ctlmsk [lnk$sv_shr] then signal (lin$optignshr);  
: 1669      1756 2   if .lnk$gl_ctlmsk [lnk$sv_sys] then signal (lin$optignsys);  
: 1670      1757 2   return true  
: 1671      1758 1   end;  
                                !Of set_stack
```

```
                                001C 00000 SET_STACK:  
                                .WORD      Save R2,R3,R4  
54 00000000G 00 9E 00002      MOVAB     LNK$GW_STACK, R4      : 1743  
53 00000000G 00 9E 00009      MOVAB     LIB$SIGNAL, R3  
52 00000000G 00 9E 00010      MOVAB     LNK$GL_CTLMSK, R2  
50          1C AC D0 00017      MOVL     28(AP), R0      : 1752  
64          50 B0 0001B      MOVW     R0, LNK$GW_STACK  
0000FFFF 8F 50 D1 0001E      CML     R0, #65535  
          04 1A 00025      BGTRU     1$  
          64 B5 00027      TSTW     LNK$GW_STACK  
          16 12 00029      BNEQ     2$  
          7E FFFF 8F 3C 0002B 1$: MOVZWL   #65535, -(SP)      : 1754  
          01 DD 00030      PUSHL    #1  
          5C DD 00032      PUSHL    AP  
          00000000' EF 9F 00034      PUSHAB   STACK_NAME  
09 00000000V 04 FB 0003A      CALLS    #4, OPTIONVALUERR  
          62 02 E1 00041 2$: BBC       #2, LNK$GL_CTLMSK, 3$      : 1755  
          00000000G 8F DD 00045      PUSHL    #LIN$ OPTIGNSHR  
          63 01 FB 0004B      CALLS    #1, LIB$SIGNAL  
09          62 03 E1 0004E 3$: BBC       #3, LNK$GL_CTLMSK, 4$      : 1756  
          00000000G 8F DD 00052      PUSHL    #LIN$ OPTIGNSYS  
          63 01 FB 00058      CALLS    #1, LIB$SIGNAL  
          50 01 D0 0005B 4$: MOVL     #1, R0      : 1757  
                                04 0005E      RET      : 1758
```

; Routine Size: 95 bytes, Routine Base: \$CODE\$ + 075C


```
: 1673      1759 1 routine definesymbolname =
: 1674      1760 2   begin
: 1675      1761 3   :
: 1676      1762 4   : This routine is called by TPARSE when the symbol name is parsed for the SYMBOL= option
: 1677      1763 5   :
: 1678      1764 6   builtin
: 1679      1765 7   ap;
: 1680      1766 8   map
: 1681      1767 9   ap : ref bblock;
: 1682      1768 10  local
: 1683      1769 11  crfst;
: 1684      1770 12  namelengthcheck (ap [tpa$l_tokencnt], lnk$gt_symstring); !Check symbol name length
: 1685      1771 13  search_insert_symbol (ap [tpa$l_tokencnt], cursymdesc, !Insert into symbol table
: 1686      1772 14  cursymsnb);
: 1687      1773 15  cursymdesc [sym$w_flags] = (.cursymdesc [sym$w_flags] and !Clear all the flag bits
: 1688      1774 16  (gsy$m_uni or sym$m_supres)) ! except uni and supres
: 1689      1775 17  or (gsy$m_def or sym$m_optsym); ! and set def and optsym
: 1690      1776 18
: 1691      1777 19  if .lnk$gl_ctlmsk [lnk$v_cros] !If cross referencing
: 1692      1778 20  then
: 1693      1779 21  begin
: 1694      1780 22  crossref_symbol (.cursymdesc, .cursymsnb, !Insert symbol definition
: 1695      1781 23  .cursymdesc [sym$w_flags]);
: 1696      1782 24  crfst = crf$insrtref (lnk$al_syblfmt, cursymsnb [snb$b_namlng], ! then insert a reference
: 1697      1783 25  defined_by_option, gsy$m_def, ! saying defined by option
: 1698      1784 26  crf$k_def); ! and this is a defining reference
: 1699      1785 27
: 1700      1786 28  if not .crfst !Report error from cref if so
: 1701      1787 29  then
: 1702      1788 30  signal (lin$_crferr, 0, .crfst);
: 1703      1789 31
: 1704      1790 32  lnk$gl_maxsymsz = maxu (.lnk$gl_maxsymsz, !Set maximum symbol name length
: 1705      1791 33  .cursymsnb [snb$b_namlng]);
: 1706      1792 34  lnk$gl_maxmodsz = maxu (.lnk$gl_maxmodsz, ! and maximum module name length
: 1707      1793 35  %charcount ('<Linker option>'));
: 1708      1794 36  lnk$gw_ncrosrfs = .lnk$gw_ncrosrfs + 1; !Count the cross reference
: 1709      1795 37  end;
: 1710      1796 38  return true
: 1711      1797 39  end; !Of definesymbolname
```

```
001C 00000 DEFINESYMBOLNAME:
      54 00000000G 00 9E 00002 .WORD Save R2,R3,R4 : 1759
      53 00000000G 00 9E 00009 MOVAB LNK$GL_MAXMODSZ, R4 :
      52 00000000' EF 9E 00010 MOVAB LNK$GL_MAXSYMSZ, R3 :
      00000000G 00 9F 00017 PUSHAB CURSYMSNB, R2 :
      10 AC 9F 0001D PUSHAB LNK$GT_SYMSTRING : 1770
00000000V EF 02 FB 00020 CALLS #2, NAMELENGTHCHECK :
      FC 52 DD 00027 PUSHL R2 : 1771
      10 AC 9F 00029 PUSHAB CURSYMDESC :
      00000000V EF 10 AC 9F 0002C PUSHAB 16(AP) :
      50 FC 03 FB 0002F CALLS #3, SEARCH_INSERT_SYMBOL :
      AC A2 D0 00036 MOVL CURSYMDESC, R0 : 1773
```

		51	0A	A0	3C	0003A	MOVZWL	10(R0), R1	1774
		51	FFFFDFFB	8F	CA	0003E	BICL2	#-8197, R1	
	0A	51	0202	8F	A9	00045	BISW3	#514, R1, 10(R0)	1775
			00000000G	00	95	0004C	TSTB	LNK\$GL_CFLMSK	1777
		7E		0A	A0	3C	BGEQ	4\$	
					62	DD	MOVZWL	10(R0), -(SP)	1781
					50	DD	PUSHL	CURSYMSNB	1780
	00000000V	EF			03	FB	PUSHL	R0	
					01	DD	CALLS	#3, CROSSREF_SYMBOL	
					02	DD	PUSHL	#1	1782
			00000000'		EF	9F	PUSHL	#2	
	7E	62			04	C1	PUSHAB	DEFINED BY OPTION	
			00000000G		00	9F	ADDL3	#4, CURSYMSNB, -(SP)	
		00			05	FB	PUSHAB	LNK\$AL_SYTBLFMT	
		11			50	E8	CALLS	#5, CRF\$INSRTREF	
					50	DD	BLBS	CRFSTS, 1\$	1786
					7E	D4	PUSHL	CRFSTS	1788
			00000000G		8F	DD	CLRL	-(SP)	
	00000000G	00			03	FB	PUSHL	#LINS CRFERR	
		50			62	D0	CALLS	#3, LIB\$SIGNAL	
		51			63	D0	MOVL	CURSYMSNB, R0	1791
51	04	08			00	ED	MOVL	LNK\$GL_MAXSYMSZ, R1	
					04	1B	CMPZV	#0, #8, 4(R0), R1	
		51	04		A0	9A	BLEQU	2\$	
		63			51	D0	MOVZBL	4(R0), R1	
		50			64	D0	MOVL	R1, LNK\$GL_MAXSYMSZ	1790
		0F			50	D1	MOVL	LNK\$GL_MAXMODSZ, R0	1792
					03	1E	CPL	R0, #15	
		50			0F	D0	BGEQU	3\$	
		64			50	D0	MOVL	#15, R0	
			00000000G		00	B6	MOVL	R0, LNK\$GL_MAXMODSZ	
		50			01	D0	INCW	LNK\$GW_NCR\$SRFS	1794
					04	000BE	MOVL	#1, R0	1796
							RET		1797

; Routine Size: 191 bytes, Routine Base: \$CODE\$ + 07BB


```
1713 1798 1 routine definesymbolval =
1714 1799 2 begin
1715 1800 3
1716 1801 4 This routine is called by TPARSE when the value is parsed in the SYMBOL= option
1717 1802 5
1718 1803 6 builtin
1719 1804 7 remque,
1720 1805 8 ap;
1721 1806 9
1722 1807 10 map
1723 1808 11 ap : ref bblock;
1724 1809 12 local
1725 1810 13 ntsymnt,
1726 1811 14 crfst;
1727 1812 15
1728 1813 16 if .cursymdesc[sym$l_udflink] neq 0 ! Is symbol on undefined list already?
1729 1814 17 then ! (via a previous UNIV=this_symbol)
1730 1815 18 begin
1731 1816 19 remque(cursymdesc[sym$l_udflink], ntsymnt); ! If so, then remove it and lower
1732 1817 20 lnsk$gw_nudfsyms = .lnk$gw_nudfsyms - 1; ! the count of undefined symbols
1733 1818 21 end;
1734 1819 22 cursymdesc [sym$l_value] = .ap [tpa$l_number]; !Set the symbol value
1735 1820 23 if .lnk$gl_ctlmsk[lnk$v_long] !If generating a long map
1736 1821 24 then
1737 1822 25 begin
1738 1823 26 crfst = crf$insrtref (lnk$al_valctlb, cursymdesc [sym$l_value],
1739 1824 27 ! then insert a reference to its value
1740 1825 28 cursymnb [snb$b_namlng], .cursymdesc [sym$w_flags], 0);
1741 1826 29 if not .crfst then signal ((lin$crferr, 0, .crfst));
1742 1827 30 end;
1743 1828 31 return true
1828 1 end; !of definesymbolval
```

```
0004 00000 DEFINESYMBOLVAL:
52 00000000' EF 9E 00002 .WORD Save R2
50 62 D0 00009 MOVAB CURSYMDESC, R2
60 D5 0000C MOVL CURSYMDESC, R0
09 13 0000E TSTL (R0)
51 60 0F 00010 BEQL 1$
00000000G 00 B7 00013 REMQUE (R0), NXTSYMNT
50 62 D0 00019 1$: DECW LNK$GW_NUDFSYMS
60 1C AC D0 0001C MOVL CURSYMDESC, R0
2E 00000000G 00 E9 00020 MOVL 28(AP), (R0)
7E D4 00027 BLBC LNK$GL_CTLMSK+1, 2$
7E 0A A0 3C 00029 CLRL -(SP)
7E 04 A2 04 C1 0002D MOVZWL 10(R0), -(SP)
00000000G 00 DD 00032 ADDL3 #4, CURSYMSNB, -(SP)
00000000G 00 9F 00034 PUSHL R0
00 05 FB 0003A PUSHAB LNK$AL_VALCTLB
11 50 E8 00041 CALLS #5, CRF$INSRTREF
50 DD 00044 BLBS CRFSTS, 2$
7E D4 00046 PUSHL CRFSTS
00000000G 8F DD 00048 CLRL -(SP)
PUSHL #LIN$_CRFERR
```

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

K 3
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 100
(30)

00000000G 00
50

03 FB 0004E
01 D0 00055 2\$: CALLS #3, LIB\$SIGNAL
04 00058 MOVL #1, R0
RET

:
: 1827
: 1828

; Routine Size: 89 bytes, Routine Base: \$CODE\$ + 087A


```
: 1745      1829 1 routine set_unsupported =
: 1746      1830 2   begin
: 1747      1831 2   ---
: 1748      1832 2   |
: 1749      1833 2   | This routine is called by TPARSE to process the UNSUPPORTED= option.
: 1750      1834 2   |
: 1751      1835 2   | Currently, only the low bit of this mask is defined -- when set it means
: 1752      1836 2   | that demand zero pages should be allowed for a shareable image. In the future,
: 1753      1837 2   | this should be done by looking at the ISD attributes, but this is too risky
: 1754      1838 2   | at this stage in the release. (The real problem is that the image-activator
: 1755      1839 2   | does not know how to deal with shared demand zero sections.)
: 1756      1840 2   |
: 1757      1841 2   | Inputs:
: 1758      1842 2   |
: 1759      1843 2   |         AP                Points to tparse block
: 1760      1844 2   |         AP[tpa$l_number]    Bit mask for LNK$GL_UNSUPPORTED
: 1761      1845 2   |
: 1762      1846 2   | Outputs:
: 1763      1847 2   |
: 1764      1848 2   |         lnk$gl_unsupported    set up
: 1765      1849 2   |
: 1766      1850 2   | ---
: 1767      1851 2   | builtin
: 1768      1852 2   |     ap;
: 1769      1853 2   |     map
: 1770      1854 2   |         ap : ref bblock;
: 1771      1855 2   |         lnk$gl_unsupported = .ap [tpa$l_number] ;
: 1772      1856 2   |         return true
: 1773      1857 1   |     end;
```

```
0000 00000 SET_UNSUPPORTED:
00000000' EF      1C  AC  D0 00002      .WORD Save nothing
50          01  D0 0000A      MOVL 28(AP), LNK$GL_UNSUPPORTED
04 0000D      RET          #1, R0
```

; Routine Size: 14 bytes, Routine Base: \$CODE\$ + 08D3

; 1774 1858 1

: 1829
: 1855
: 1856
: 1857

```
: 1776      1859  1 routine set_universal =
: 1777      1860  2   begin
: 1778      1861  2   :
: 1779      1862  2   : This routine is called for each symbol in the UNIVERSAL option by TPARSE
: 1780      1863  2   :
: 1781      1864  2   builtin
: 1782      1865  2   ap;
: 1783      1866  2   map
: 1784      1867  2   ap : ref bblock;
: 1785      1868  2
: 1786      1869  2   namelengthcheck (ap [tpa$l_tokencnt], lnk$gt_symstring); !Check length of symbol name
: 1787      1870  2   search_insert_symbol (ap [tpa$l_tokencnt], cursymdesc, cursymsnb);
: 1788      1871  2   !Lookup/insert symbol into symbol table
: 1789      1872  2   cursymdesc [sym$w_flags] = (.cursymdesc[sym$w_flags] and ! Preserve def and supres
: 1790      1873  2   (gsy$m_def or sym$m_supres)) ! bits, and set universal
: 1791      1874  2   or gsy$m_uni; ! flag
: 1792      1875  2   if not .cursymdesc[sym$v_def] ! If not defined,
: 1793      1876  2   then ! then insert in
: 1794      1877  2   lnk$insudfsym (.cursymdesc); ! undefined list
: 1795      1878  2   crossref_symbol (.cursymdesc, .cursymsnb, !Cross reference the symbol
: 1796      1879  2   .cursymdesc[sym$w_flags]);
: 1797      1880  2   return true
: 1798      1881  1   end; !Of set_univ
```

```
0004 00000 SET_UNIVERSAL:
      52 00000000' EF 9E 00002 .WORD Save R2 : 1859
      00000000G 00 9F 00009 MOVAB CURSYMDESC, R2 : 1869
      10 AC 9F 0000F PUSHAB LNK$GT_SYMSTRING
00000000V EF 02 FB 00012 PUSHAB 16(AP)
      04 A2 9F 00019 CALLS #2, NAMELENGTHCHECK : 1870
      10 52 DD 0001C PUSHAB CURSYMSNB
      AC 9F 0001E PUSHAB 16(AP)
00000000V EF 03 FB 00021 CALLS #3, SEARCH_INSERT_SYMBOL
      50 62 D0 00028 MOVL CURSYMDESC, R0 : 1872
      51 0A A0 3C 0002B MOVZWL 10(R0), R1 : 1873
      51 FFFDFDFD 8F CA 0002F BICL2 #-8195, R1
      0A A0 09 0A A0 01 E0 0003B BISW3 #4, R1, 10(R0) : 1874
      0A A0 01 E0 0003B BBS #1, 10(R0), 1$ : 1875
      00000000G 00 50 DD 00040 PUSHL R0 : 1877
      50 01 FB 00042 CALLS #1, LNK$INSUDFSYM
      7E 0A A0 3C 0004C 1$: MOVL CURSYMDESC, R0 : 1879
      04 A2 DD 00050 MOVZWL 10(R0), -(SP)
      50 DD 00053 PUSHL CURSYMSNB : 1878
00000000V EF 03 FB 00055 CALLS #3, CROSSREF_SYMBOL
      50 01 D0 0005C MOVL #1, R0 : 1880
      04 0005F RET : 1881
```

; Routine Size: 96 bytes, Routine Base: \$CODE\$ + 08E1

LNK_PROCOPTIONS Linker options parser
V04=000 TPARSE action routines

N 3
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 103
(33)

```
: 1800      1882 1 routine set_alluniv =  
: 1801      1883 2   begin  
: 1802      1884 2   |  
: 1803      1885 2   | Flag all universal temporarily  
: 1804      1886 2   |  
: 1805      1887 2   |   lnk$gl_ctlmsk [lnk$u_alluniv] = true;  
: 1806      1888 2   |   return true;  
: 1807      1889 1   end;
```

```
0000 00000 SET_ALLUNIV:  
00000000G 00      04 88 00002 .WORD Save nothing  
50          01 D0 00009   BISB2 #4, LNK$GL_CTLMSK+3  
          04 0000C   MOVL #1, R0  
          RET
```

```
: 1882  
: 1887  
: 1888  
: 1889
```

; Routine Size: 13 bytes, Routine Base: \$CODE\$ + 0941

```

: 1809      1890 1 routine set_include =
: 1810      1891 2   begin
: 1811      1892 2   |
: 1812      1893 2   | This routine processes the /include qualifier
: 1813      1894 2   |
: 1814      1895 2   |   builtin
: 1815      1896 2   |   ap;
: 1816      1897 2   |   map
: 1817      1898 2   |   ap : ref bblock;
: 1818      1899 2   |   set_qual_flags (.ap, (fdb$m_libr or fdb$m_libextr));   !Set qualifier flag
: 1819      1900 2   |   lnk$setlibrin(1);
: 1820      1901 2   |   return true
: 1821      1902 1   end;
                                     !Of set_include

```

				0000 00000 SET_INCLUDE:			
	7E	42	8F 9A 00002	.WORD	Save nothing	:	1890
			5C DD 00006	MOVZBL	#66, -(SP)	:	1899
00000000V	EF		02 FB 00008	PUSHL	AP	:	
			01 DD 0000F	CALLS	#2, SET_QUAL_FLAGS	:	
00000000G	00		01 FB 00011	PUSHL	#1	:	1900
	50		01 DO 00018	CALLS	#1, LNK\$SETLIBRIN	:	1901
			04 0001B	MOVL	#1, R0	:	1902
				RET		:	

; Routine Size: 28 bytes, Routine Base: \$CODE\$ + 094E


```

: 1823      1903 1 routine set_library =
: 1824      1904 2   begin
: 1825      1905 2   |
: 1826      1906 2   | This routine processes the /library qualifier
: 1827      1907 2   |
: 1828      1908 2   builtin
: 1829      1909 2   ap;
: 1830      1910 2   map
: 1831      1911 2   ap : ref bblock;
: 1832      1912 2   set_qual_flags (.ap, (fdb$m_libr or fdb$m_libsrch));
: 1833      1913 2   lnk$setlibrin(1);
: 1834      1914 2   return true
: 1835      1915 1   end;
                                !Of set_library

```

				0000 00000 SET_LIBRARY:			
	7E	82	8F 9A 00002	.WORD	Save nothing	: 1903	
			5C DD 00006	MOVZBL	#130, -(SP)	: 1912	
00000000V	EF		02 FB 00008	PUSHL	AP	:	
			01 DD 0000F	CALLS	#2, SET_QUAL_FLAGS	: 1913	
00000000G	00		01 FB 00011	PUSHL	#1	:	
	50		01 DO 00018	CALLS	#1, LNK\$SETLIBRIN	: 1914	
			04 0001B	MOVL	#1, R0	: 1915	
				RET			

; Routine Size: 28 bytes, Routine Base: \$CODE\$ + 096A

```

1837      1916      1 routine set_selective =
1838      1917      2     begin
1839      1918      2     !
1840      1919      2     ! This routine processes the /selective qualifier
1841      1920      2     !
1842      1921      2     builtin
1843      1922      2     ap:
1844      1923      2     map
1845      1924      2     ap : ref bblock;
1846      1925      2     set_qual_flags (.ap, fdb$m_selser);
1847      1926      2     return true
1848      1927      1     end;

```

		0000 00000	SET_SELECTIVE:			
			.WORD	Save nothing		: 1916
	08	DD 00002	PUSHL	#8		: 1925
	5C	DD 00004	PUSHL	AP		: :
00000000V	02	FB 00006	CALLS	#2, SET_QUAL_FLAGS		: :
EF	01	D0 0000D	MOVL	#1, R0		: 1926
50		04 00010	RET			: 1927

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 0986


```
: 1850      1928 1 routine set_shareable =
: 1851      1929 2   begin
: 1852      1930 2   |
: 1853      1931 2   | This routine processes the /shareable qualifier
: 1854      1932 2   |
: 1855      1933 2   |   builtin
: 1856      1934 2   |   ap;
: 1857      1935 2   |   map
: 1858      1936 2   |   ap : ref bblock;
: 1859      1937 2   |   local
: 1860      1938 2   |   descr : bblock [dsc$c_s_bln];
: 1861      1939 2   |
: 1862      1940 2   |   set_qual_flags (.ap, fdb$m_shr);
: 1863      1941 2   |   if .lnk$gl_ctlmsk [[lnk$v_sys]           !Don't allow /share with /sys
: 1864      1942 2   |   then
: 1865      1943 2   |       signal_stop (lin$shrinsys);
: 1866      1944 2   |
: 1867      1945 2   |   | If this is in a cluster= option and there are other
: 1868      1946 2   |   | files in the cluster, issue a warning and create a new
: 1869      1947 2   |   | cluster.
: 1870      1948 2   |   |
: 1871      1949 2   |   | if .cluoptflag and .lnk$gl_curclu [clu$lfstfdb] neq 0
: 1872      1950 2   |   | then
: 1873      1951 2   |   |   begin
: 1874      1952 2   |   |   descr [dsc$w_length] = .tparse_block [tpa$l_tokenptr] - .cmdbuffer;
: 1875      1953 2   |   |   descr [dsc$a_pointer] = .cmdbuffer;
: 1876      1954 2   |   |   signal (lin$shrsepclu, 0, lin$optlin, 3, descr, tparse_block [tpa$l_tokencnt],
: 1877      1955 2   |   |   tparse_block [tpa$l_stringcnt]);
: 1878      1956 2   |   |   lnk$alloccluster (lnk$gl_curclu);           !Allocate a new cluster
: 1879      1957 2   |   |   lnk$gl_curclu [clu$v_protect] = .protectflag; !Set protect if on
: 1880      1958 2   |   |   end;
: 1881      1959 2   |   |
: 1882      1960 2   |   | if not .cluoptflag           !If not processing cluster= option
: 1883      1961 2   |   | then
: 1884      1962 2   |   |   begin
: 1885      1963 2   |   |   lnk$alloccluster (lnk$gl_curclu);       ! then allocate a new cluster descriptor
: 1886      1964 2   |   |   lnk$gl_curclu [clu$v_protect] = .protectflag; !Set protect if on
: 1887      1965 2   |   |   end;
: 1888      1966 2   |   |
: 1889      1967 2   |   | lnk$gl_curclu [clu$v_shring] = true;
: 1890      1968 2   |   | lnk$gl_curclu [clu$v_copy] = .sharecopy;
: 1891      1969 2   |   | lnk$setshrblin ();
: 1892      1970 2   |   | return true
: 1893      1971 1   | end;           !Of set_shareable
```

001C 00000 SET_SHAREABLE:

54	00000000G	00	9E	00002	.WORD	Save R2,R3,R4
53	00000000G	00	9E	00009	MOVAB	LNK\$ALLOCLUSTER, R4
52	00000000G	EF	9E	00010	MOVAB	LNK\$GL_CURCLU, R3
5E		08	C2	00017	MOVAB	CLUOPTFLAG, R2
		04	DD	0001A	SUBL2	#8, SP
		5C	DD	0001C	PUSHL	#4
					PUSHL	AP

```
: 1928
:
:
:
: 1940
:
```

PC	OP	RS	RT	RD	RS	RT	RD	OP	RS	RT	RD	PC
00000000V	EF	02	FB	0001E	CALLS	#2, SET QUAL FLAGS						
00000000G	00	03	E1	00025	BBC	#3, LNK\$GL CTLM\$K, 1\$						1941
		08F	DD	0002D	PUSHL	#LIN\$ SHRINSYS						1943
00000000G	00	01	FB	00033	CALLS	#1, LIB\$STOP						
	45	62	E9	0003A	BLBC	CLUOPTFLAG, 3\$						1949
	50	63	D0	0003D	MOVL	LNK\$GL_CURCLU, R0						
		08	A0	D5 00040	TSTL	8(R0)						
			3A	13 00043	BEQL	2\$						
6E	B4	A2	C4	A2 A3 00045	SUBW3	CMDBUFFER, TPARSE_BLOCK+20, DESCR						1952
	04	AE	C4	A2 D0 0004B	MOVL	CMDBUFFER, DESCR+4						1953
			A8	A2 9F 00050	PUSHAB	TPARSE_BLOCK+8						1955
			B0	A2 9F 00053	PUSHAB	TPARSE_BLOCK+16						1954
			08	AE 9F 00056	PUSHAB	DESCR						
			03	DD 00059	PUSHL	#3						
			8F	DD 0005B	PUSHL	#LIN\$_OPTLIN						
			7E	D4 00061	CLRL	-(SP)						
			8F	DD 00063	PUSHL	#LIN\$ SHRSEPCLU						
00000000G	00	07	FB	00069	CALLS	#7, LIB\$SIGNAL						1956
			53	DD 00070	PUSHL	R3						
			01	FB 00072	CALLS	#1, LNK\$ALLOCLUSTER						1957
			63	D0 00075	MOVL	LNK\$GL_CURCLU, R0						
58	A0	01	08	A2 F0 00078	INSV	PROTECTFLAG, #7, #1, 88(R0)						
			62	E8 0007F	BLBS	CLUOPTFLAG, 4\$						1960
			53	DD 00082	PUSHL	R3						1963
			01	FB 00084	CALLS	#1, LNK\$ALLOCLUSTER						
			63	D0 00087	MOVL	LNK\$GL_CURCLU, R0						1964
58	A0	01	08	A2 F0 0008A	INSV	PROTECTFLAG, #7, #1, 88(R0)						
			63	D0 00091	MOVL	LNK\$GL_CURCLU, R0						1967
			04	88 00094	BISB2	#4, 88(R0)						
58	A0	01	04	A2 F0 00098	INSV	SHARECOPY, #4, #1, 88(R0)						1968
			00	FB 0009F	CALLS	#0, LNK\$SETSHRBLIN						1969
			01	D0 000A6	MOVL	#1, R0						1970
			04	000A9	RET							1971

; Routine Size: 170 bytes, Routine Base: \$CODES + 0997

LNK
V04


```

: 1895      1972 1 routine set_shrcopyflag =
: 1896      1973 2   begin
: 1897      1974 2   |
: 1898      1975 2   | This routine is called by TPARSE to set/clear the copyflag for /share
: 1899      1976 2   |
: 1900      1977 2   |   builtin
: 1901      1978 2   |   ap;
: 1902      1979 2   |   map
: 1903      1980 2   |   ap : ref bblock;
: 1904      1981 2   |   if .sharecopy neq 0
: 1905      1982 2   |   then
: 1906      1983 2   |   |   begin
: 1907      1984 2   |   |   signal (lin$ shrcpyign);
: 1908      1985 2   |   |   sharecopy = 0;
: 1909      1986 2   |   |   end;
: 1910      1987 2   |   return true
: 1911      1988 1   end;

```

!Of set_shrcopyflag

				0004 00000 SET_SHRCOPYFLAG:		
	52	00000000'	EF 9E 00002	.WORD	Save R2	: 1972
			62 D5 00009	MOVAB	SHARECOPY, R2	: 1981
			0F 13 0000B	TSTL	SHARECOPY	: 1984
			8F DD 0000D	BEQL	1\$	: 1985
00000000G	00	00000000G	01 FB 00013	PUSHL	#LINS SHRCOPYIGN	: 1987
			62 D4 0001A	CALLS	#1, LIB\$SIGNAL	: 1988
	50		01 D0 0001C	CLRL	SHARECOPY	
			04 0001F	MOVL	#1, R0	
				RET		

; Routine Size: 32 bytes, Routine Base: \$CODE\$ + 0A41

```

: 1913      1989 1 routine insertcluster =
: 1914      1990 2   begin
: 1915      1991 2   |
: 1916      1992 2   | This routine inserts the current cluster in the cluster tree
: 1917      1993 2   |
: 1918      1994 2   |   local
: 1919      1995 2   |   blockaddr;
: 1920      1996 2   |
: 1921      1997 2   |   if lib$lookup_tree (lnk$gl_clutree, lnk$gl_curclu [clu$b_namlng],   !If cluster name already in tree
: 1922      1998 2   |       lnk$clunamcmp, blockaddr)
: 1923      1999 2   |   then
: 1924      2000 2   |       signal (lin$mulcluopty, 2, lnk$gl_curclu [clu$b_namlng],       ! then tell user
: 1925      2001 2   |       .optionfilename);
: 1926      2002 2   |
: 1927      2003 2   |   lnk$insert_clu (.lnk$gl_curclu);           !Insert the cluster in the tree
: 1928      2004 2   |   return true
: 1929      2005 1   | end;

```

		0004 00000 INSERTCLUSTER:				
	52	00000000G	00	9E	00002	Save R2
	5E		04	C2	00009	LNK\$GL_CURCLU, R2
			5E	DD	0000C	#4, SP
		00000000G	00	9F	0000E	SP
7E	62	0000005C	8F	C1	00014	LNK\$CLUNAMCMP
		00000000G	00	9F	0001C	#92, LNK\$GL_CURCLU, -(SP)
	00		04	FB	00022	LNK\$GL_CLUTREE
	1D		50	E9	00029	#4, LIB\$LOOKUP_TREE
		00000000'	EF	DD	0002C	R0, 1\$
7E	62	0000005C	8F	C1	00032	OPTIONFILENAME
			02	DD	0003A	#92, LNK\$GL_CURCLU, -(SP)
		00000000G	8F	DD	0003C	#2
	00		04	FB	00042	#LINS_MULCLUOPT
			62	DD	00049	#4, LIB\$SIGNAL
	00		01	FB	0004B	LNK\$GL_CURCLU
	50		01	D0	00052	#1, LNK\$INSERT_CLU
			04	00	00055	#1, R0
						RET

; Routine Size: 86 bytes, Routine Base: \$CODE\$ + 0A61


```
1931 2006 1 routine processfile =
1932 2007 2   begin
1933 2008 2   |
1934 2009 2   | This routine is called by TPARSE when a complete file specification
1935 2010 2   | has been scanned.
1936 2011 2   |
1937 2012 2   builtin
1938 2013 2   ap;
1939 2014 2   map
1940 2015 2   ap : ref bblock;
1941 2016 2   local
1942 2017 2   descr : bblock [dsc$c_s_bln];
1943 2018 2   bind
1944 2019 2   filebits = .fileflagsadr : bitvector;
1945 2020 2
1946 2021 2   if (.filebits and fdb$m_libr)      ! Call lnk$setlibrin now that all
1947 2022 2   then                               ! qualifier parsing is done
1948 2023 2   lnk$setlibrin(1);
1949 2024 2   if .cluoptflag                     !If processing a cluster= option
1950 2025 2   then
1951 2026 2   begin
1952 2027 2   if .lnk$gl_curclu [clu$v_shrimg]    !If current cluster is shareable image
1953 2028 2   and ((.filebits and fdb$m_shr) eql 0) ! and this file is not a shareable image
1954 2029 2   then
1955 2030 2   begin
1956 2031 2   descr [dsc$w_length] = .tparse_block [tpa$l_tokenptr] - .cmdbuffer;
1957 2032 2   descr [dsc$a_pointer] = .cmdbuffer; !Create string descriptor for part of line
1958 2033 2   signal (lin$shrsepclu, 0,          !Tell user we are creating new cluster
1959 2034 2   lin$opt[in, 3, descr, tparse_block [tpa$l_tokencnt], tparse_block [tpa$l_stringcnt]);
1960 2035 2   lnk$alloccluster (lnk$gl_curclu); !Allocate a new cluster
1961 2036 2   descr [dsc$w_length] = sym$c_maxlng; !Set length of buffer
1962 2037 2   descr [dsc$a_pointer] = lnk$gl_curclu [clu$t_name]; !And starting address
1963 2038 2   sys$fao (uplit (stringdesc ('!AC_!UL')), descr, descr,
1964 2039 2   lnk$gl_defclu [clu$b_namlng], .lnk$gl_defclunum); !Create cluster name of default_cluster_N
1965 2040 2   lnk$gl_curclu [clu$b_namlng] = .descr [dsc$w_length]; !Set resulting cluster name length
1966 2041 2   lnk$gl_defclunum = .lnk$gl_defclunum + 1;
1967 2042 2   lnk$gl_curclu [clu$v_protect] = .protectflag;
1968 2043 2   end;
1969 2044 2   end
1970 2045 2   else
1971 2046 2   if .lnk$gl_curclu [clu$v_shrimg]    !Not cluster option, is current cluster shr image?
1972 2047 2   and .lnk$gl_curclu [clu$l_fstfdb] neq 0 ! and there are files in the cluster
1973 2048 2   then
1974 2049 2   lnk$gl_curclu = lnk$gl_defclu; !Put this file in the default cluster
1975 2050 2
1976 2051 2   | If this file is a shareable image, or the first file in a CLUSTER=
1977 2052 2   | option, then insert the cluster into the cluster list
1978 2053 2   |
1979 2054 2   | if .lnk$gl_curclu [clu$v_shrimg] or (.cluoptflag and (.lnk$gl_curclu [clu$l_fstfdb] eql 0))
1980 2055 2   | then
1981 2056 2   | begin
1982 2057 2   | if .lnk$gl_curclu [clu$v_shrimg]    !If cluster is shareable image
1983 2058 2   | then
1984 2059 2   | begin
1985 2060 2   | getfilename (filespec_desc, descr); ! then we must set the cluster name
1986 2061 2   | getfilename (filespec_desc, descr); !Get filename portion of file spec
1987 2062 2   |
```

```
: 1988      2063  4      lnk$gl_curclu [clu$b namlng] = .descr [dsc$w length];      !Set length into cluster descriptor
: 1989      2064  4      ch$move (.descr [dsc$w length], .descr [dsc$a_pointer],      !Copy in cluster name
: 1990      2065  4      lnk$gl_curclu [clu$t_name]);
: 1991      2066      end;
: 1992      2067      insertcluster ();      !Insert cluster in cluster tree
: 1993      2068      end;
: 1994      2069
: 1995      2070      lnk$gl_inclst = include_desc;
: 1996      2071      inputfile(filespec_desc, 1);
: 1997      2072      return true
: 1998      2073  1      end;      !Of processfile
```

```
.PSECT $SPLITS,NOWRT,NOEXE,2
```

```
00 4C 55 21 5F 43 41 21 00088 P.AAN: .ASCII \!AC_!UL\<0>
                                00000007 00090 P.AAM: .LONG 7
                                00000000' 00094 .ADDRESS P.AAN
```

```
.PSECT $CODE$,NOWRT,2
```

```
01FC 00000 PROCESSFILE:
```

```
58 00000000' EF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8      : 2006
57 00000000G 00 9E 00009 MOVAB LNK$GL_DEFCLUNUM, R8
56 00000000' EF 9E 00010 MOVAB LNK$GL_CURCLU, R7
5E      08 C2 00017 MOVAB CLUOPTFLAG, R6
52      D0 A6 D0 0001A SUBL2 #8, SP
50      67 D0 0001E MOVL FILEFLAGSADR, R2
76      66 E9 00021 MOVL LNK$GL_CURCLU, R0
7E      02 E1 00024 BLBC CLUOPTFLAG, 1$
58      A0 02 E1 00024 BBC #2, 88(R0), 1$
62      62 02 E0 00029 BBS #2, (R2), 2$
7E      C4 A6 A3 0002D SUBW3 CMDBUFFER, TPARSE_BLOCK+20, DESCR
6E      B4 A6 D0 00033 MOVL CMDBUFFER, DESCR+4
      04 AE A6 9F 00038 PUSHAB TPARSE_BLOCK+8
      C4 A6 9F 0003B PUSHAB TPARSE_BLOCK+16
      08 AE 9F 0003E PUSHAB DESCR
      03 DD 00041 PUSHL #3
      00000000G 8F DD 00043 PUSHL #LINS_OPTLIN
      00000000G 7E D4 00049 CLRL -(SP)
      00000000G 8F DD 0004B PUSHL #LINS_SHRSEPCLU
      00000000G 00 07 FB 00051 CALLS #7, LIB$SIGNAL
      00000000G 00 57 DD 00058 PUSHL R7
      00000000G 00 01 FB 0005A CALLS #1, LNK$ALLOCLUSTER
      6E      1F B0 00061 MOVW #31, DESCR
      04 AE 67 000005D 8F C1 00064 ADDL3 #93, LNK$GL_CURCLU, DESCR+4
      00000000G 68 DD 0006D PUSHL LNK$GL_DEFCLUNUM
      08 AE 9F 0006F PUSHAB LNK$GL_DEFCLU+92
      0C AE 9F 00075 PUSHAB DESCR
      00000000' EF 9F 00078 PUSHAB DESCR
      00000000G 00 05 FB 0007B PUSHAB P.AAM
      50      67 D0 00081 CALLS #5, SYSS$FAO
      5C A0 6E 90 0008B MOVL LNK$GL_CURCLU, R0
      6E 90 0008B MOVW DESCR, -92(R0)
```


58	A0	01	07	08	68	D6	0008F	INCL	LNK\$GL_DEFCLUNUM	:	2043
					A6	F0	00091	INSV	PROTECTFLAG, #7, #1, 88(R0)	:	2044
		0C	58	A0	11	11	00098	BRB	2\$	:	2024
				08	02	E1	0009A	BBC	#2, 88(R0), 2\$	:	2048
					A0	D5	0009F	TSTL	8(R0)	:	2049
					07	13	000A2	BEQL	2\$	:	
			67	00000000G	00	9E	000A4	MOVAB	LNK\$GL_DEFCLU, LNK\$GL_CURCLU	:	2051
			50		67	D0	000AB	MOVL	LNK\$GL_CURCLU, R0	:	2056
		0D	58	A0	02	E0	000AE	BBS	#2, 88(R0), 3\$	:	
				28	66	E9	000B3	BLBC	CLUOPTFLAG, 5\$	:	
					A0	D5	000B6	TSTL	8(R0)	:	
				08	23	12	000B9	BNEQ	5\$	:	
		19	58	A0	02	E1	000BB	BBC	#2, 88(R0), 4\$	:	2059
					5E	DD	000C0	PUSHL	SP	:	2062
				F0	A6	9F	000C2	PUSHAB	FILESPEC_DESC	:	
			00000000V	EF	02	FB	000C5	CALLS	#2, GETFILENAME	:	
				50	67	D0	000CC	MOVL	LNK\$GL_CURCLU, R0	:	2063
			5C	A0	6E	90	000CF	MOVB	DESCR, -92(R0)	:	
5D	A0		04	BE	6E	28	000D3	MOVC3	DESCR, @DESCR+4, 93(R0)	:	2065
			FECC	CF	00	FB	000D9	CALLS	#0, INSERTCLUSTER	:	2067
			FC	A8	A6	9E	000DE	MOVAB	INCLUDE_DESC, LNK\$GL_INCLST	:	2070
					01	DD	000E3	PUSHL	#1	:	2071
				F0	A6	9F	000E5	PUSHAB	FILESPEC_DESC	:	
			00000000G	00	02	FB	000E8	CALLS	#2, INPUTFILE	:	2072
				50	01	D0	000EF	MOVL	#1, R0	:	2073
					04	00	000F2	RET		:	

; Routine Size: 243 bytes, Routine Base: \$CODE\$ + 0AB7

```
: 2000      2074 1 routine getfilename (filedesc, retdesc) =
: 2001      2075   begin
: 2002      2076   |
: 2003      2077   | This routine returns the filename portion of a
: 2004      2078   | file specification
: 2005      2079   |
: 2006      2080   map
: 2007      2081     filedesc : ref bblock,
: 2008      2082     retdesc  : ref bblock;
: 2009      2083   local
: 2010      2084     status,
: 2011      2085     rslasc : vector [2],
: 2012      2086     rslbuf : bblock [nam$c_maxrss],
: 2013      2087     rsabuf : bblock [nam$c_maxrss],
: 2014      2088     esabuf : bblock [nam$c_maxrss],
: 2015      2089     filefab : bblock [fab$c_bln],
: 2016      2090     filename : bblock [nam$c_bln];
: 2017      2091
: 2018      2092     rslasc [0] = nam$c_maxrss;
: 2019      2093     rslasc [1] = rslbuf;
: 2020      2094
: 2021      2095   if (status = $strlog (lognam = .filedesc, ! If the filedesc is a logical name,
: 2022      2096       rslbuf = rslasc)) eql ss$_normal
: 2023      2097   then
: 2024      2098       begin
: 2025      2099         retdesc [dsc$w_length] = .filedesc [dsc$w_length];
: 2026      2100         retdesc [dsc$a_pointer] = .filedesc [dsc$a_pointer];
: 2027      2101         return true;
: 2028      2102       end;
: 2029      2103
: 2030      2104   $nam_init (nam = filename, rsa = rsabuf, rss = nam$c_maxrss, esa = esabuf, ess = nam$c_maxrss);
: 2031      2105   $fab_init (fab = filefab, fop = nam, fna = .filedesc [dsc$a_pointer], fns = .filedesc [dsc$w_length],
: 2032      2106       nam = filename);
: 2033      2107   $parse (fab = filefab);
: 2034      2108   retdesc [dsc$w_length] = .filename [nam$b_name];
: 2035      2109   retdesc [dsc$a_pointer] = ch$find_sub (.filedesc [dsc$w_length], .filedesc [dsc$a_pointer],
: 2036      2110       .filename [nam$b_name], .filename [nam$l_name]);
: 2037      2111   return true
: 2038      2112   end;
```

.EXTRN SYS\$STRNLOG, SYS\$PARSE

007C 0000 GETFILENAME:

	SE	FC48	CE	9E	00002	.WORD	Save R2,R3,R4,R5,R6	: 2074
						MOVAB	-952(SP), SP	
F8	AD	FF	8F	9A	00007	MOVZBL	#255, RSLDSC	: 2092
FC	AD	FEF8	CD	9E	0000C	MOVAB	RSLBUF, RSLDSC+4	: 2093
			7E	7C	00012	CLRQ	-(SP)	: 2096
			7E	D4	00014	CLRL	-(SP)	
		F8	AD	9F	00016	PUSHAB	RSLDSC	
			7E	D4	00019	CLRL	-(SP)	
	56	04	AC	D0	0001B	MOVL	FILEDESC, R6	
			56	DD	0001F	PUSHL	R6	
00000000G	00		06	FB	00021	CALLS	#6, SYS\$STRNLOG	
	01		50	D1	00028	CMPL	STATUS, #1	

0060	8F	00	04	50	08	0E	12	0002B	BNEQ	1\$		
				60		AC	D0	0002D	MOVL	RETDESC, R0	2099	
				A0	04	66	B0	00031	MOVW	(R6), (R0)		
						A6	D0	00034	MOVL	4(R6), 4(R0)	2100	
						79	11	00039	BRB	3\$	2101	
				6E		00	2C	0003B	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	2104	
						6E		00042				
				6E	6002	8F	B0	00043	MOVW	#24578, \$RMS_PTR		
				02		01	8E	00048	MNEGB	#1, \$RMS_PTR+2		
				04		CE	9E	0004C	MOVAB	RSABUF, \$RMS_PTR+4		
				0A		01	8E	00052	MNEGB	#1, \$RMS_PTR+10		
				0C		CE	9E	00056	MOVAB	ESABUF, \$RMS_PTR+12		
0050	8F	00		6E		00	2C	0005C	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	2106	
						AE		00063				
				60		8F	B0	00065	MOVW	#20483, \$RMS_PTR		
				64	5003	8F	D0	0006B	MOVL	#16777216, \$RMS_PTR+4		
				76	01000000	02	90	00073	MOVB	#2, \$RMS_PTR+22		
				7F		02	90	00077	MOVB	#2, \$RMS_PTR+31		
				0088		6E	9E	0007B	MOVAB	FILENAM, \$RMS_PTR+40		
				008C		A6	D0	00080	MOVL	4(R6), \$RMS_PTR+44		
				0094		66	90	00086	MOVB	(R6), \$RMS_PTR+52		
						AE	9F	0008B	PUSHAB	FILEFAB	2107	
				00000000G		01	FB	0008E	CALLS	#1, SYSSPARSE		
				54		AC	D0	00095	MOVL	RETDESC, R4	2108	
				64		AE	9B	00099	MOVZBW	FILENAM+59, (R4)		
				55		AE	9A	0009D	MOVZBL	FILENAM+59, R5	2110	
04	B6	66		BE		55	39	000A1	MATCHC	R5, @FILENAM+76, (R6), @4(R6)		
						03	13	000A8	BEQL	2\$		
				53		55	D0	000AA	MOVL	R5, R3		
				53		55	C2	000AD	SUBL2	R5, R3		
				04		53	D0	000B0	MOVL	R3, 4(R4)		
				50		01	D0	000B4	MOVL	#1, R0	2111	
						04	000B7	RET			2112	

; Routine Size: 184 bytes, Routine Base: \$CODE\$ + 0BAA

```
: 2040      2113 1 routine set_qual_flags (tpablock, qualbits) =
: 2041      2114 1
: 2042      2115 1 This routine checks for conflicting qualifiers and then sets the proper bits
: 2043      2116 1
: 2044      2117 2 begin
: 2045      2118 2 map
: 2046      2119 2 tpablock : ref bblock;
: 2047      2120 2 local
: 2048      2121 2 descr : bblock [dsc$c_s_bln];
: 2049      2122 2
: 2050      2123 2 if (..fileflagsadr and .tpablock [tpa$l_param]) neq 0 !Check for conflicting qualifiers
: 2051      2124 2 then
: 2052      2125 2 begin
: 2053      2126 2 descr [dsc$w_length] = .tpablock [tpa$l_tokenptr] - .cndbuffer;
: 2054      2127 2 descr [dsc$a_pointer] = .cndbuffer;
: 2055      2128 2 signal_stop (lin$ confqual, 0, lin$ optlin, 3, descr, tpablock [tpa$l_tokencnt],
: 2056      2129 2 tpablock [tpa$l_stringcnt]);
: 2057      2130 2 end
: 2058      2131 2 else
: 2059      2132 2 .fileflagsadr = ..fileflagsadr or .qualbits;
: 2060      2133 2 return true
: 2061      2134 1 end; !Of set_qual_flags
```

				0004 00000 SET_QUAL_FLAGS:					
		52	00000000'	EF	9E	00002	WORD	Save R2	2113
		5E		08	C2	00009	MOVAB	FILEFLAGSADR, R2	
		50	04	AC	D0	0000C	SUBL2	#8, SP	
	20	A0	00	B2	D3	00010	MOVL	TPABLOCK, R0	2123
				2D	13	00015	BITL	@FILEFLAGSADR, 32(R0)	
							BEQL	1\$	
6E	14	A0	F4	A2	A3	00017	SUBW3	CMDBUFFER, 20(R0), DESCR	2126
	04	AE	F4	A2	D0	0001D	MOVL	CMDBUFFER, DESCR+4	2127
			08	A0	9F	00022	PUSHAB	8(R0)	2129
			10	A0	9F	00025	PUSHAB	16(R0)	2128
			08	AE	9F	00028	PUSHAB	DESCR	
				03	DD	0002B	PUSHL	#3	2129
			00000000G	8F	DD	0002D	PUSHL	#LINS_OPTLIN	
				7E	D4	00033	CLRL	-(SP)	
			00000000G	8F	DD	00035	PUSHL	#LINS_CONFQUAL	
	00000000G	00		07	FB	0003B	CALLS	#7, LIB\$STOP	
				05	11	00042	BRB	2\$	2123
	00	B2	08	AC	C8	00044	BISL2	QUALBITS, @FILEFLAGSADR	2132
		50		01	D0	00049	MOVL	#1, R0	2133
				04	0004C		RET		2134

; Routine Size: 77 bytes, Routine Base: \$CODE\$ + 0C62


```
2063 2135 1 %sbttl 'Search / insert into linked, ordered list';
2064 2136 1 routine search_insert_list (listhead, namedesc, blocksize, blockaddr) =
2065 2137 2 begin
2066 2138 2 ---
2067 2139 2 This routine searches a linked list for a given name. If the name is
2068 2140 2 not found, a new node is created and the name is inserted. The structure
2069 2141 2 of the links in the list must be as follows:
2070 2142 2
2071 2143 2 ptr==> link to next entry
2072 2144 2 size of name (byte)
2073 2145 2 name
2074 2146 2 (any other data)
2075 2147 2
2076 2148 2 Inputs:
2077 2149 2
2078 2150 2 listhead the address of the linked list head
2079 2151 2 blocksize size of block to allocate if not found
2080 2152 2
2081 2153 2 Outputs:
2082 2154 2
2083 2155 2 blockaddr entry address
2084 2156 2
2085 2157 2 ---
2086 2158 2 macro
2087 2159 2 nextblock = 0,0,32,0%, !First longword is link
2088 2160 2 namesize = 4,0,8,0%, !Then a byte of name length
2089 2161 2 nameaddr = 5,0,0,0%; !Followed by name
2090 2162 2
2091 2163 2 map
2092 2164 2 listhead : ref bblock,
2093 2165 2 namedesc : ref bblock;
2094 2166 2 local
2095 2167 2 lastblock : ref bblock,
2096 2168 2 thisblock : ref bblock;
2097 2169 2
2098 2170 2 thisblock = .listhead; !Start at top of list
2099 2171 2 lastblock = .thisblock;
2100 2172 2
2101 2173 2 Loop looking for entry, or one less than desired entry
2102 2174 2
2103 2175 2 while (thisblock = .thisblock [nextblock]) neq 0 do
2104 2176 2 if ch$eq1 (.thisblock [namesize], thisblock [nameaddr], .namedesc [dsc$w_length],
2105 2177 2 .namedesc [dsc$a_pointer], 0)
2106 2178 2 then
2107 2179 2 begin
2108 2180 2 .blockaddr = .thisblock; !Return found entry to caller
2109 2181 2 return true;
2110 2182 2 end
2111 2183 2 else
2112 2184 2 lastblock = .thisblock;
2113 2185 2
2114 2186 2 Name was not found...allocate a new one
2115 2187 2
2116 2188 2 lnk$alloblk (.blocksize, thisblock);
2117 2189 2 ch$fill (0, .blocksize, .thisblock); !Zero the block
2118 2190 2 thisblock [nextblock] = .lastblock [nextblock]; !Link into the list
2119 2191 2 lastblock [nextblock] = .thisblock;
2119 2191 2 thisblock [namesize] = .namedesc [dsc$w_length];
```

```
: 2120      2192  2      ch$move (.namedesc [dsc$w_length], .namedesc [dsc$a_pointer], thisblock [nameaddr]);
: 2121      2193  2      .blockaddr = .thisblock;
: 2122      2194  2      return true
: 2123      2195  1      end;

                                !Of search_insert_list
```

```
                                01FC 00000 SEARCH_INSERT_LIST:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8
                                PUSH  LISTHEAD
                                04 AC DD 00002 PUSH  THISBLOCK, LASTBLOCK
                                58 6E D0 00005 MOV  THISBLOCK, R4
                                54 6E D0 00008 MOV  NAMEDESC, R7
                                57 08 AC D0 0000B MOV  (R4), THISBLOCK
                                6E 64 D0 0000F 1$: BEQ  3$
                                1D 13 00012 MOV  THISBLOCK, R4
                                54 6E D0 00014 MOVZBL 4(R4), R0
                                50 04 A4 9A 00017 CMPC5  R0, 5(R4), #0, @NAMEDESC, @4(R7)
                                08 BC 00 05 A4 50 2D 0001B
                                04 B7 00 00022 BNEQ  2$
                                06 06 12 00024 MOV  R4, @BLOCKADDR
                                54 54 D0 00026 BRB  4$
                                31 11 0002A MOV  R4, LASTBLOCK
                                58 54 D0 0002C 2$: BRB  1$
                                DE 11 0002F 3$: PUSH  SP
                                5E DD 00031 PUSH  BLOCKSIZE
                                0C AC DD 00033 CALLS #2, LNK$ALLOBLK
                                02 02 FB 00036 MOV  THISBLOCK, R6
                                6E 6E D0 0003D MOV  #0, (SP), #0, BLOCKSIZE, (R6)
                                00 00 2C 00040
                                66 66 00046 MOV  (LASTBLOCK), (R6)
                                68 68 D0 00047 MOV  R6, (LASTBLOCK)
                                56 56 D0 0004A MOV  @NAMEDESC, 4(R6)
                                08 BC 90 0004D MOV  @NAMEDESC, @4(R7), 5(R6)
                                08 BC 28 00052 MOV  R6, @BLOCKADDR
                                56 56 D0 00059 MOV  #1, R0
                                01 01 D0 0005D 4$: RET
                                04 04 00060
```

; Routine Size: 97 bytes, Routine Base: \$CODE\$ + 0CAF


```
2125 2196 1 %sbttl 'Search / insert a symbol';
2126 2197 1 routine search_insert_symbol (symboldesc, symblock, snblock) =
2127 2198 2 begin
2128 2199 2 ---
2129 2200 2 This routine looks up the symbol in the symbol table and returns
2130 2201 2 the address of the symbol block. If it is not found, it is inserted.
2131 2202 2
2132 2203 2 Inputs:
2133 2204 2
2134 2205 2     symboldesc    Address of a string descriptor for symbol name
2135 2206 2     symblock      Address of location to return symbol block address
2136 2207 2     snblock       Address of location to return symbol name block address
2137 2208 2
2138 2209 2 ---
2139 2210 2 map
2140 2211 2     symboldesc : ref bblock;
2141 2212 2 local
2142 2213 2     symptr : ref bblock,
2143 2214 2     symbolstring : vector [sym$cmxlng*2, byte];
2144 2215 2     ch$move ((symbolstring [0] = .symboldesc [dsc$wlength]), !Create an ASCII string
2145 2216 2     .symboldesc [dsc$apointer], symbolstring [1]);
2146 2217 2     if lnk$search (symbolstring, .symblock, .snblock) !Look up the symbol
2147 2218 2     then
2148 2219 2         return true;                ! and if found then all done
2149 2220 2
2150 2221 2 Not in table, so insert it
2151 2222 2
2152 2223 2     lnk$insert (symbolstring, symptr, .snblock);
2153 2224 2     symptr [sym$l_udflink] = 0;
2154 2225 2     symptr [sym$w_flags] = 0;
2155 2226 2     .symblock = .symptr;
2156 2227 2     return true
2157 2228 1 end;                                !Of search_insert_list
```

```
003C 00000 SEARCH_INSERT_SYMBOL:
05 AE 04 04 7E 08 AC 7D 00017 MOVQ Save R2,R3,R4,R5
00000000G 00 0C AE 9F 0001B MOVAB -68(SP), SP
1C 0C 04 AE 9F 0002B MOVAB SYMBOLDESC, R0
00000000G 00 0C AE 9F 0002E MOVZWL (R0), R1
08 BC 50 D0 00040 MOVAB R1, SYMBOLSTRING
00000000G 00 0C AE 9F 0002E MOVAB R1, @4(R0), SYMBOLSTRING+1
08 BC 50 D0 00040 MOVQ SYMBOLBLOCK, -(SP)
00000000G 00 0C AE 9F 0002E PUSHAB SYMBOLSTRING
08 BC 50 D0 00040 CALLS #3, LNK$SEARCH
00000000G 00 0C AE 9F 0002E BLBS R0, 1$
08 BC 50 D0 00040 PUSHL SNBLOCK
00000000G 00 0C AE 9F 0002E PUSHAB SYMPTR
08 BC 50 D0 00040 CALLS #3, LNK$INSERT
00000000G 00 0C AE 9F 0002E MOVAB SYMPTR, R0
08 BC 50 D0 00040 CLRL (R0)
00000000G 00 0C AE 9F 0002E CLRW 10(R0)
08 BC 50 D0 00040 MOVAB R0, @SYMBOLBLOCK
```

LNK_PROCOPTIONS Linker options parser
V04=000 Search / insert a symbol

E 5
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 120
(44)

50

01 D0 00044 1\$: MOVL #1, R0
04 00047 RET

: 2227
: 2228

; Routine Size: 72 bytes, Routine Base: \$CODE\$ + 0D10

LNK
V04


```

2159 2229 1 %sbtcl 'Cross reference a symbol';
2160 2230 1 routine crossref_symbol (symbolblock, snblock, flags) =
2161 2231 2   begin
2162 2232 2     ---
2163 2233 2
2164 2234 2     Enter symbol into cross reference if generating one.
2165 2235 2
2166 2236 2     Inputs:
2167 2237 2
2168 2238 2         symbolblock      Pointer to symbol descriptor block
2169 2239 2         snblock          Pointer to name part of symbol block
2170 2240 2         flags            Flags to enter into cross reference
2171 2241 2
2172 2242 2     ---
2173 2243 2     map
2174 2244 2         symbolblock : ref bblock,
2175 2245 2         snblock : ref bblock;
2176 2246 2     local
2177 2247 2         crfst;
2178 2248 2
2179 2249 2     if .lnk$gl_ctlmsk [lnk$v_map]          !If generating a map
2180 2250 2     and not .lnk$gl_ctlmsk [lnk$v_brief]  ! that is not brief
2181 2251 2     then
2182 2252 2         begin
2183 2253 2             crfst = crf$insrtkey (lnk$al_sytblfmt,          ! then insert the symbol
2184 2254 2                 snblock [snb$b_namlng],                    ! passing the name
2185 2255 2                 symbolblock [sym$l_value],                  ! the value address
2186 2256 2                 .flags);                                     ! and the flags
2187 2257 2
2188 2258 2             if not .crfst then signal (lin$_crferr, 0, .crfst);
2189 2259 2
2190 2260 2             lnk$gl_maxsymsz = maxu (.lnk$gl_maxsymsz,      !Set maximum symbol name length
2191 2261 2                 .snblock [snb$b_namlng]);
2192 2262 2         end;
2193 2263 2     return true
2194 2264 2 end;

```

000C 00000 CROSSREF SYMBOL:						
	53	00000000G	00 9E 00002	WORD	Save R2,R3	: 2230
48	00		04 E1 00009	MOVAB	LNK\$GL_MAXSYMSZ, R3	: 2249
40	00		01 E0 00011	BBC	#4, LNK\$GL_CTLMSK, 3\$	: 2250
		0C	AC DD 00019	BBS	#1, LNK\$GL_CTLMSK+1, 3\$	: 2256
		04	AC DD 0001C	PUSHL	FLAGS	: 2255
	52	08	AC D0 0001F	PUSHL	SYMBOLBLOCK	: 2254
		04	A2 9F 00023	MOVL	SNBLOCK, R2	: 2253
		00000000G	00 9F 00026	PUSHAB	4(R2)	: 2255
00000000G	00		04 FB 0002C	PUSHAB	LNK\$AL_SYTBLEMT	: 2258
	11		50 E8 00033	CALLS	#4, CRF\$INSRTKEY	: 2258
			50 DD 00036	BLBS	CRFSTS, 1\$	: 2258
			7E D4 00038	PUSHL	CRFSTS	: 2258
		00000000G	8F DD 0003A	CLRL	-(SP)	: 2258
00000000G	00		03 FB 00040	PUSHL	#LIN\$ CRFERR	: 2258
				CALLS	#3, LIB\$SIGNAL	: 2258

50	04	A2	50
			08
			50
			63
			50

	63	DO	00047	1\$:
	00	ED	0004A	
	04	1B	00050	
04	A2	9A	00052	
	50	DO	00056	2\$:
	01	DO	00059	3\$:
		04	0005C	

```

MOVL      LNK$GL_MAXSYMSZ, R0
CMPZV     #0, #8, 4(R2), R0
BLEQU     2$
MOVZBL    4(R2), R0
MOVL      R0, LNK$GL_MAXSYMSZ
MOVL      #1, R0
RET

```

: 2261
:
:
:
:
:
: 2260
: 2263
: 2264

; Routine Size: 93 bytes, Routine Base: \$CODES + 0D58

LNK_PROCOPTIONS
V04=000

Linker options parser
Debug routine to catch TPARSE in action

H 5
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 123
(46)

```
: 2196      2265 1 %sbttl 'Debug routine to catch TPARSE in action';
: 2197      2266 1 routine debug_stop =
: 2198      2267 2   begin
: 2199      2268 2   return true
: 2200      2269 1   end;
```

0000 00000 DEBUG_STOP:

50

01 D0 00002
04 00005

.WORD
MOVL
RET

Save nothing
#1, R0

: 2266
: 2268
: 2269

; Routine Size: 6 bytes, Routine Base: \$CODE\$ + 0DB5

```
: 2202      2270 1 %sbttl 'Check length of name';
: 2203      2271 1 routine namelengthcheck (namedesc, nametype, special_length) =
: 2204      2272 2   begin
: 2205      2273 2   |
: 2206      2274 2   | This routine checks that the symbol name is gtr 0
: 2207      2275 2   | and leq sym$sc_maxlng and issues a SIGNAL_STOP if not
: 2208      2276 2   |
: 2209      2277 2   |   map
: 2210      2278 2   |     namedesc : ref bblock;
: 2211      2279 2   |     local
: 2212      2280 2   |       max_length;
: 2213      2281 2   |     builtin
: 2214      2282 2   |       nullparameter;
: 2215      2283 2   |
: 2216      2284 2   |     if not nullparameter (3)
: 2217      2285 2   |       then max_length = .special_length
: 2218      2286 2   |       else max_length = sym$sc_maxlng;
: 2219      2287 2   |     if .namedesc [dsc$w_length] eql 0 or .namedesc [dsc$w_length] gtru .max_length
: 2220      2288 2   |       !Check for illegal name length
: 2221      2289 2   |     then
: 2222      2290 2   |       begin
: 2223      2291 2   |         local
: 2224      2292 2   |           symbolbuf : vector [512, byte];      !Local buffer to copy name
: 2225      2293 2   |
: 2226      2294 2   |           symbolbuf [0] = .namedesc [dsc$w_length];      !Create ASCII string
: 2227      2295 2   |           ch$move (minu (.namedesc [dsc$w_length], 512), !Copy as much of name in as possible
: 2228      2296 2   |             .namedesc [dsc$a_pointer], symbolbuf [1]);
: 2229      2297 2   |           signal_stop (lin$il[namelen, 6, .nametype,      !Report error and quit
: 2230      2298 2   |             symbolbuf, .namedesc [dsc$w_length], .max_length, optionfilestring, .optionfilename);
: 2231      2299 2   |         end;
: 2232      2300 2   |     return true
: 2233      2301 1   end;
```

00FC 0000 NAMELENGTHCHECK:

					.WORD	Save R2,R3,R4,R5,R6,R7	2271
5E	FE00	CE	9E	00002	MOVAB	-512(SP), SP	
03		6C	91	00007	CMPB	(AP), #3	2284
		0B	1F	0000A	BLSSU	1\$	
	0C	AC	D5	0000C	TSTL	12(AP)	
		06	13	0000F	BEQL	1\$	
57	0C	AC	D0	00011	MOVL	SPECIAL_LENGTH, MAX_LENGTH	2285
		03	11	00015	BRB	2\$	
57		1F	D0	00017	MOVL	#31, MAX_LENGTH	2286
56	04	AC	D0	0001A	MOVL	NAMEDESC, R6	2287
		66	B5	0001E	TSTW	(R6)	
		07	13	00020	BEQL	3\$	
57	66	00	ED	00022	CMPZV	#0, #16, (R6), MAX_LENGTH	
		3E	1B	00027	BLEQU	5\$	
6E		66	90	00029	MOVB	(R6), SYMBOLBUF	2294
50		66	3C	0002C	MOVZWL	(R6), R0	2295
0200	8F	50	B1	0002F	CMPW	R0, #512	
		05	1B	00034	BLEQU	4\$	
50	0200	8F	3C	00036	MOVZWL	#512, R0	

LNK_PROCOPTIONS Linker options parser
V04=000 Check length of name

J 5
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 125
(47)

01	AE	04	B6	00000000'	50	28	0003B	4\$:	MOV C3	R0, @4(R6), SYMBOLBUF+1	:	2296
				00000000'	EF	DD	00041		PUSHL	OPTIONFILENAME	:	2298
					EF	9F	00047		PUSHAB	OPTIONFILESTRING	:	2297
			7E		57	DD	0004D		PUSHL	MAX_LENGTH	:	2298
					66	3C	0004F		MOVZWL	(R6), -(SP)	:	
		10			AE	9F	00052		PUSHAB	SYMBOLBUF	:	2297
		08			AC	DD	00055		PUSHL	NAMETYPE	:	
					06	DD	00058		PUSHL	#6	:	
				00000000G	8F	DD	0005A		PUSHL	#LINS_ILLNAMELEN	:	
			00		08	FB	00060		CALLS	#8, LIB\$STOP	:	
			50		01	DC	00067	5\$:	MOVL	#1, R0	:	2300
					04	0006A			RET		:	2301

; Routine Size: 107 bytes, Routine Base: \$CODE\$ + 0DBB

LNK
V04

```
: 2235      2302 1 %sbttl 'Options parsing error routines';
: 2236      2303 1 routine syntaxerr =
: 2237      2304 2   begin
: 2238      2305 2   :
: 2239      2306 2   : This routine is called by TPARSE to report a syntax error and quit
: 2240      2307 2   :
: 2241      2308 2   local
: 2242      2309 2   descr : bblock [dsc$c_s_bln];
: 2243      2310 2
: 2244      2311 2   descr [dsc$w_length] = .tparse_block [tpa$l_tokenptr] - .cmdbuffer;
: 2245      2312 2   descr [dsc$a_pointer] = .cmdbuffer;
: 2246      2313 2   signal_stop (lin$optsynerr, 1, .optionfilename, lin$optlin, 3, descr, tparse_block [tpa$l_tokencnt],
: 2247      2314 2   tparse_block [tpa$l_stringcnt]);
: 2248      2315 2   return true
: 2249      2316 1   end;
                                     !SIGNAL_STOP doesn't return, but...
                                     !Of syntaxerr
```

```
                                0004 00000 SYNTAXERR:
                                .WORD      Save R2
                                MOVAB      CMDBUFFER, R2
                                SUBL2      #8, SP
                                SUBW3      CMDBUFFER, TPARSE_BLOCK+20, DESCR
                                MOVL       CMDBUFFER, DESCR+4
                                PUSHAB     TPARSE_BLOCK+8
                                PUSHAB     TPARSE_BLOCK+16
                                PUSHAB     DESCR
                                PUSHL      #3
                                PUSHL      #LIN$ OPTLIN
                                PUSHL      OPTIONFILENAME
                                PUSHL      #1
                                PUSHL      #LIN$ OPTSYNERR
                                CALLS      #8, LIB$STOP
                                MOVL       #1, R0
                                RET
                                : 2303
                                :
                                : 2311
                                : 2312
                                : 2314
                                : 2315
                                : 2316
```

6E	F0	A2	52	00000000'	EF	9E	00002
		5E			08	C2	00009
	04	AE			62	A3	0000C
			E4		62	D0	00011
			EC		A2	9F	00015
			08		A2	9F	00018
					AE	9F	0001B
					03	DD	0001E
				00000000G	8F	DD	00020
				34	A2	DD	00026
					01	DD	00029
				00000000G	8F	DD	0002B
00000000G	00				08	FB	00031
	50				01	D0	00038
					04	0003B	

; Routine Size: 60 bytes, Routine Base: \$CODE\$ + 0E26


```
: 2251      2317 1 routine optionvaluerr (option_name, tpablock, minval, maxval) =  
: 2252      2318 2   begin  
: 2253      2319 2   :  
: 2254      2320 2   : This routine signals an option value error and stops the image.  
: 2255      2321 2   :  
: 2256      2322 2   map  
: 2257      2323 2   tpablock : ref bblock;  
: 2258      2324 2   local  
: 2259      2325 2   descr : bblock [dsc$sc_s_bln];  
: 2260      2326 2  
: 2261      2327 2   descr [dsc$w_length] = .tpablock [tpa$l_tokenptr] - .cmdbuffer;  
: 2262      2328 2   descr [dsc$a_pointer] = .cmdbuffer;  
: 2263      2329 2   signal_stop (lin$optvalerr, 5, .option_name, .tpablock [tpa$l_number], .minval, .maxval,  
: 2264      2330 2   .optionfilename, lin$optlin, 3, descr, tpablock [tpa$l_tokencnt], tpablock [tpa$l_stringcnt]);  
: 2265      2331 2   return true  
: 2266      2332 1   end;  
                                     !Of optionvaluerr
```

0004 00000 OPTIONVALUERR:						
		52	00000000'	EF 9E 00002	.WORD Save R2	: 2317
		5E		08 C2 00009	MOVAB CMDBUFFER, R2	
		50	08	AC D0 0000C	SUBL2 #8, SP	
6E	14	A0		62 A3 00010	MOVL TPABLOCK, R0	: 2327
	04	AE		62 D0 00015	SUBW3 CMDBUFFER, 20(R0), DESCR	
			08	A0 9F 00019	MOVL CMDBUFFER, DESCR+4	: 2328
			10	A0 9F 0001C	PUSHAB 8(R0)	: 2330
			08	AE 9F 0001F	PUSHAB 16(R0)	
				03 DD 00022	PUSHAB DESCR	: 2329
			00000000G	8F DD 00024	PUSHL #3	: 2330
		34	A2 DD 0002A	PUSHL #LIN\$ OPTLIN		
	7E	0C	AC 7D 0002D	PUSHL OPTIONFILENAME		
		1C	A0 DD 00031	MOVQ MINVAL, -(SP)		
		04	AC DD 00034	PUSHL 28(R0)		
			05 DD 00037	PUSHL OPTION_NAME		
			8F DD 00039	PUSHL #5		
00000000G	G0		0C FB 0003F	PUSHL #LIN\$ OPTVALERR		
	50		01 D0 00046	CALLS #12, [IB\$STOP		
			04 00049	MOVL #1, R0		: 2331
				RET		: 2332

; Routine Size: 74 bytes, Routine Base: \$CODE\$ + 0E62

```

: 2268      2333 1 routine missingargerr =
: 2269      2334 2   begin
: 2270      2335 2   |
: 2271      2336 2   | This routine is called by TPARSE to report a missing argument for an option
: 2272      2337 2   |
: 2273      2338 2   |   local
: 2274      2339 2   |     descr : bblock [dsc$c_s_bln];
: 2275      2340 2   |
: 2276      2341 2   |     descr [dsc$w_length] = .tparse_block [tpa$l_tokenptr] - .cmdbuffer;
: 2277      2342 2   |     descr [dsc$a_pointer] = .cmdbuffer;
: 2278      2343 2   |     signal_stop (lin$optargmis, 1, .optionfilename, lin$_optlin, 3, descr, tparse_block [tpa$l_tokencnt],
: 2279      2344 2   |       tparse_block [tpa$l_stringcnt]);
: 2280      2345 2   |     return true
: 2281      2346 1   end;

```

!Of missingargerr

```

                                0004 00000 MISSINGARGERR:
                                .WORD      Save R2
                                MOVAB      CMDBUFFER, R2
                                SUB! 2     #8, SP
                                SUBW3      CMDBUFFER, TPARSE_BLOCK+20, DESCR
                                MOVL       CMDBUFFER, DESCR+4
                                PUSHAB     TPARSE_BLOCK+8
                                PUSHAB     TPARSE_BLOCK+16
                                PUSHAB     DESCR
                                PUSHL      #3
                                PUSHL      #LIN$ OPTLIN
                                PUSHL      OPTIONFILENAME
                                PUSHL      #1
                                PUSHL      #LIN$ OPTARGMIS
                                CALLS      #8, LIB$STOP
                                MOVL       #1, R0
                                RET
                                6E      FO  52 00000000' EF 9E 00002
                                04      A2 5E      08 C2 00009
                                AE      62 A3 0000C
                                E4      62 D0 00011
                                EC      A2 9F 00015
                                08      A2 9F 00018
                                08      AE 9F 0001B
                                00000000G 03 DD 0001E
                                34      8F DD 00020
                                00000000G 01 DD 00026
                                00000000G 01 DD 00029
                                00      8F DD 0002B
                                50      08 FB 00031
                                01      D0 00038
                                04      04 0003B
                                .WORD      Save R2
                                MOVAB      CMDBUFFER, R2
                                SUB! 2     #8, SP
                                SUBW3      CMDBUFFER, TPARSE_BLOCK+20, DESCR
                                MOVL       CMDBUFFER, DESCR+4
                                PUSHAB     TPARSE_BLOCK+8
                                PUSHAB     TPARSE_BLOCK+16
                                PUSHAB     DESCR
                                PUSHL      #3
                                PUSHL      #LIN$ OPTLIN
                                PUSHL      OPTIONFILENAME
                                PUSHL      #1
                                PUSHL      #LIN$ OPTARGMIS
                                CALLS      #8, LIB$STOP
                                MOVL       #1, R0
                                RET

```

; Routine Size: 60 bytes, Routine Base: \$CODE\$ + 0EAC

```

: 2282      2347 1 end
: 2283      2348 1
: 2284      2349 0 eludom

```

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	44	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	108	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	152	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
_LIB\$KEYOS	112	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)

LNK_PROCOPTIONS Linker options parser
V04=000 Options parsing error routines

N 5
16-Sep-1984 00:22:56
5-Sep-1984 22:11:29

VAX-11 Bliss-32 V4.0-742
[LINKER.SRC]LNKOPTION.B32;1

Page 129
(50)

```
: LIB$STATES      1859 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1)
: -LIB$KEY1$      406 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1)
: $CODE$        3816 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
: . ABS .         0 NOVEC,NOWRT,NORD ,NOEXE,NOSHR, LCL, ABS, CON,NOPIC,ALIGN(0)
```

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	96	0	1000	00:01.8
-\$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	35	83	14	00:00.2
-\$255\$DUA28:[LINKER.OBJ]DATBAS.L32;1	538	50	9	28	00:00.8

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS\$;LNKOPTION/OBJ=OBJ\$;LNKOPTION MSRC\$;LNKOPTION/UPDATE=(ENH\$;LNKOPTION)

```
: Size:          3816 code + 2681 data bytes
: Run Time:      04:03.6
: Elapsed Time:  07:12.0
: Lines/CPU Min: 578
: Lexemes/CPU-Min: 70016
: Memory Used:  707 pages
: Compilation Complete
```


0218 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0219 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LNKPROLTB
LIS

LNKSYMTBL
LIS

LNKSYMOUT
LIS

LNKUMALLO
LIS

LNKPSCTBL
LIS

LNKPROSHR
LIS

LNKSTATSD
LIS